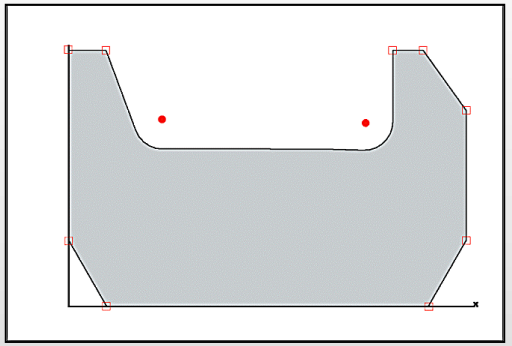
Optimal Sizing and Shape Optimization in Structural Mechanics

SFB F013 ”Symbolic and Numerical Computation”
University of Linz, Inst. for Comp.Math.

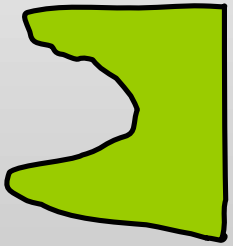
Gundolf Haase,
Michael Fischer, Christian Rathberger , Wolfram Mühlhuber

What’s that !?

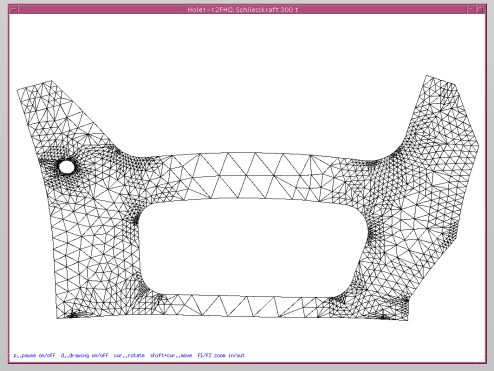
- **Shape optimization**



- **Optimal sizing (thickness)**



- **Structural Mechanics**



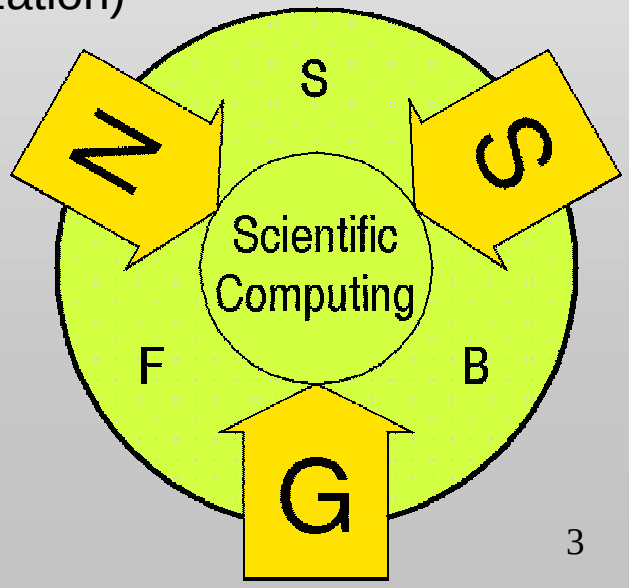
Cooperation

- **ENGEL-Group, Schwertberg(Austria), Guelph (Canada), York (USA)**

  (2D-optimization, thickness- + shape optimization)

- **SFB 013, Univ. Linz**

  (FWF, federal province OÖ, city Linz)



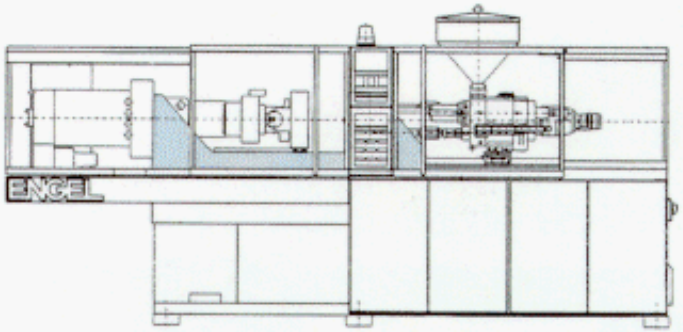
Contents

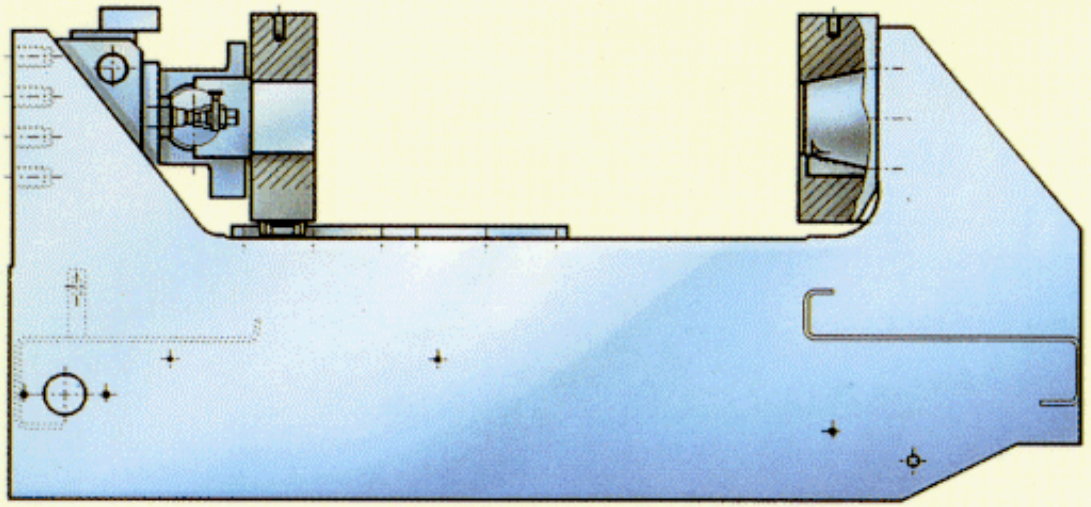
- **Shape optimization of a machine frame**
- **Optimal sizing - “ -**
- **Mathematical abstractions**
- **Gradient calculation and Geometry**

DER C-RAHMEN:

Das mechanische Kernstück einer holmlos Spritzgießmaschine.

Konstruktive Maßnahmen plus Analyse des technischen Anforderungsprofiles ist gleich patentierte Rahmenkonstruktion.

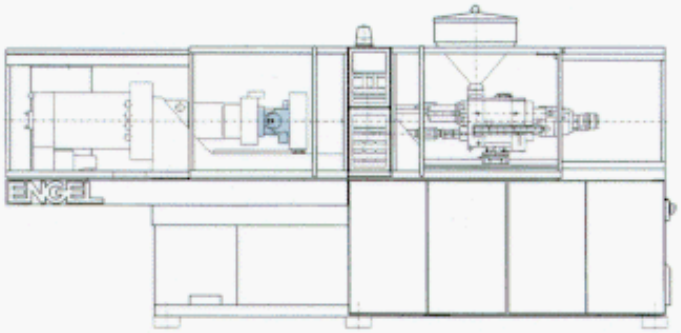


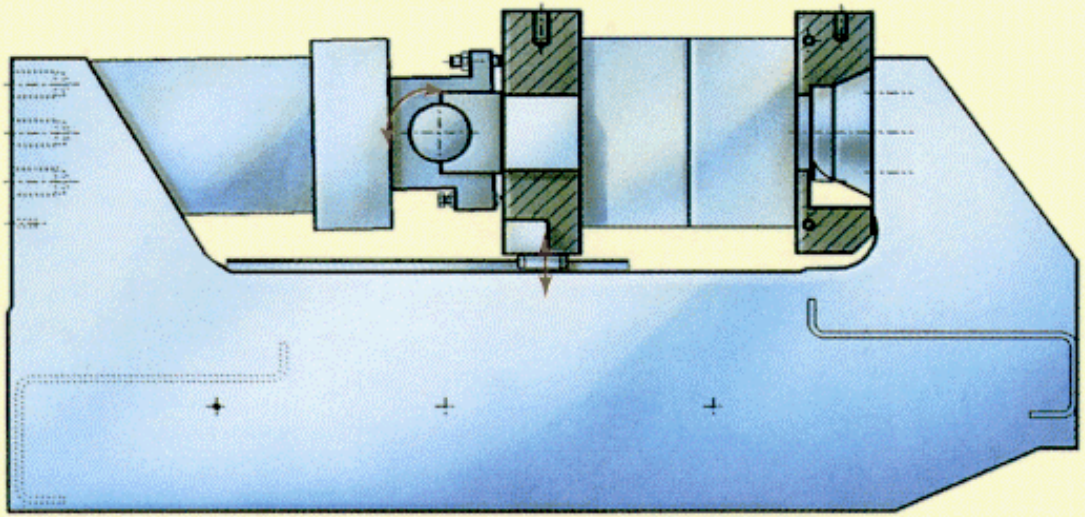


DAS HL-GELENK:

So einfach,
daß es schon wieder genial ist.

Systembedingte Verformung plus patentiertes Drehgelenk ist gleich Präzision durch selbstjustierende Plattenparallelität.



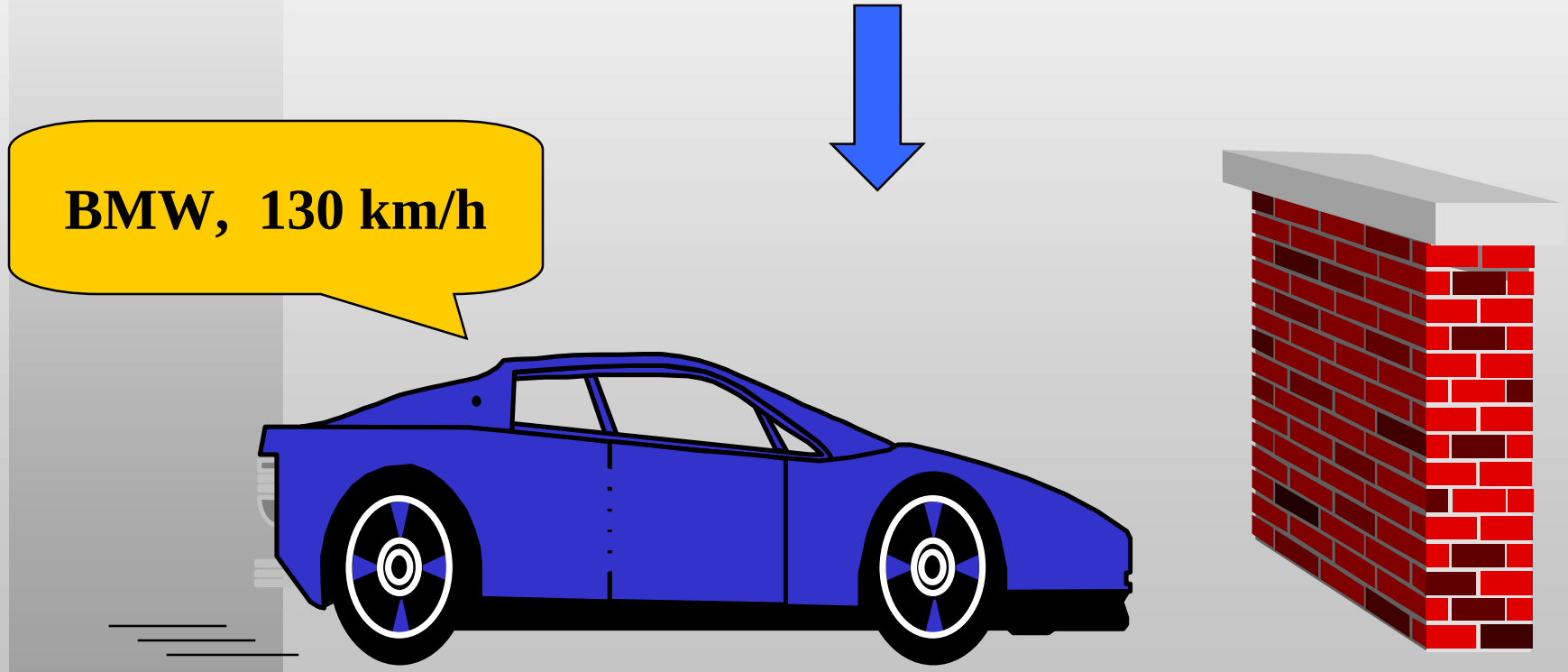


Injection moulding machine

- **Production of**
  - **plastic pieces and tools**
  - **high precision (3 gram, precision 1/100 mm)**
  - **30 work pieces per minute**
- **Clumping force to 4000 kN (~400 t)**
- **Mass to 25 tons**
- **Length to 3 meter**

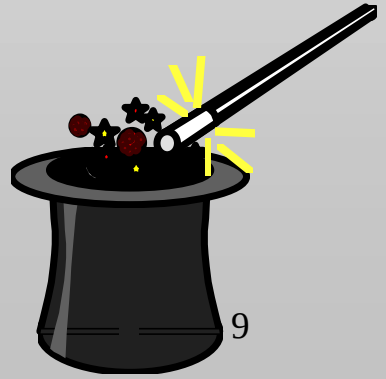
Would you like to try it ?

- **150 tons clumping force per wing**

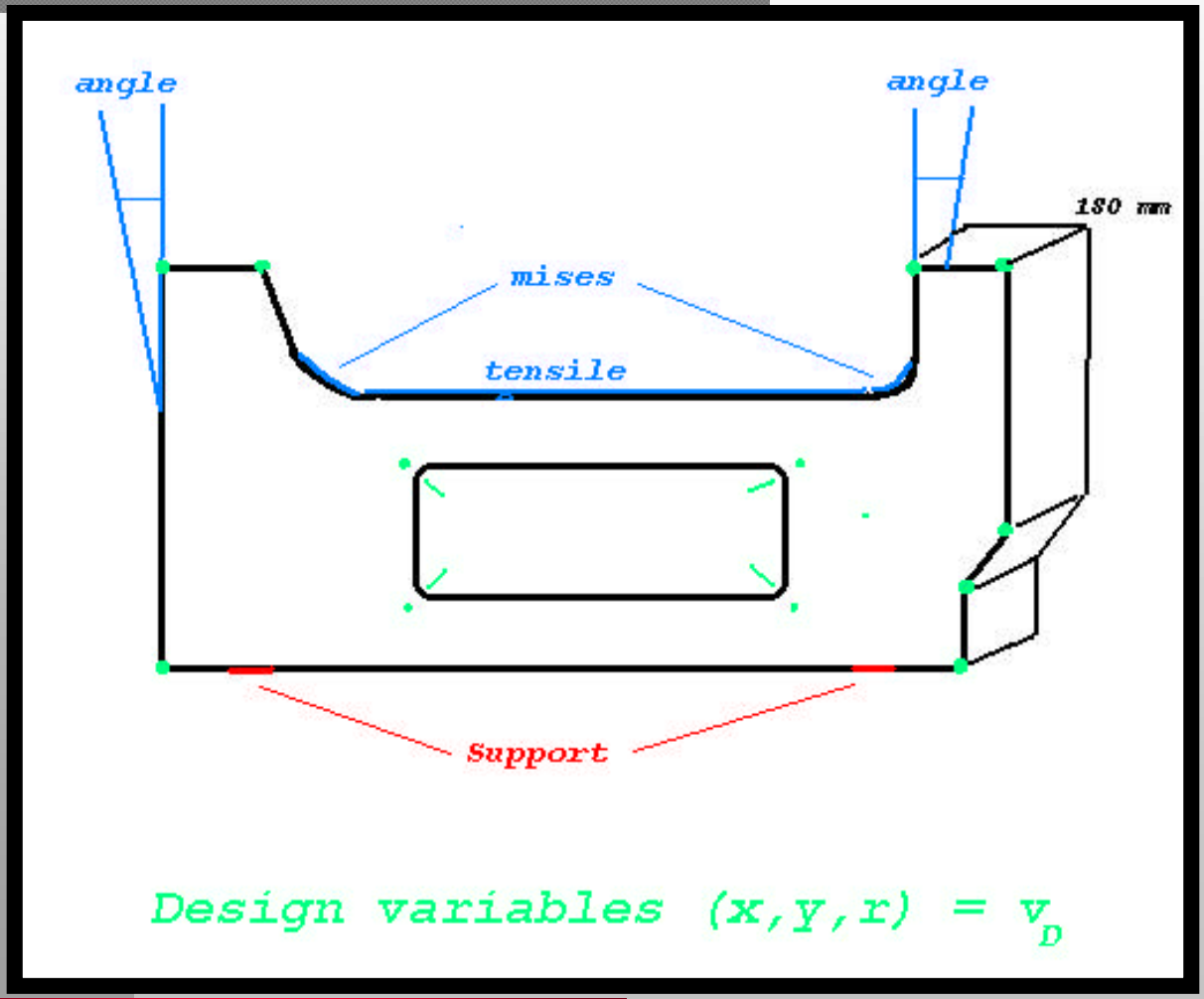


Objectives of the project

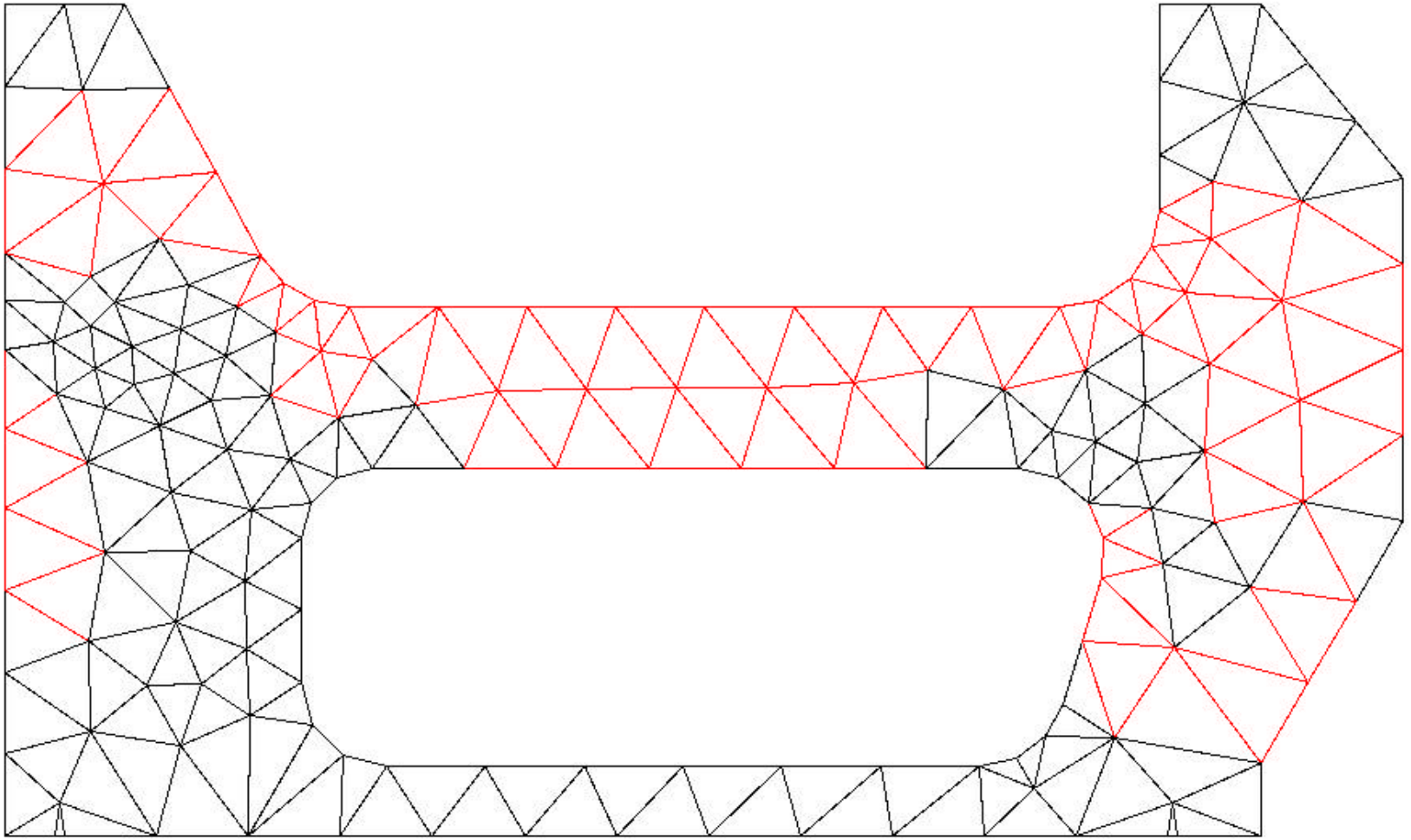
- **Mass reduction**
- **Shortening of development cycle**
- **Application of advanced math. techniques on practical problems**
- **Development of new optimization strategies**



Geometry and constraints

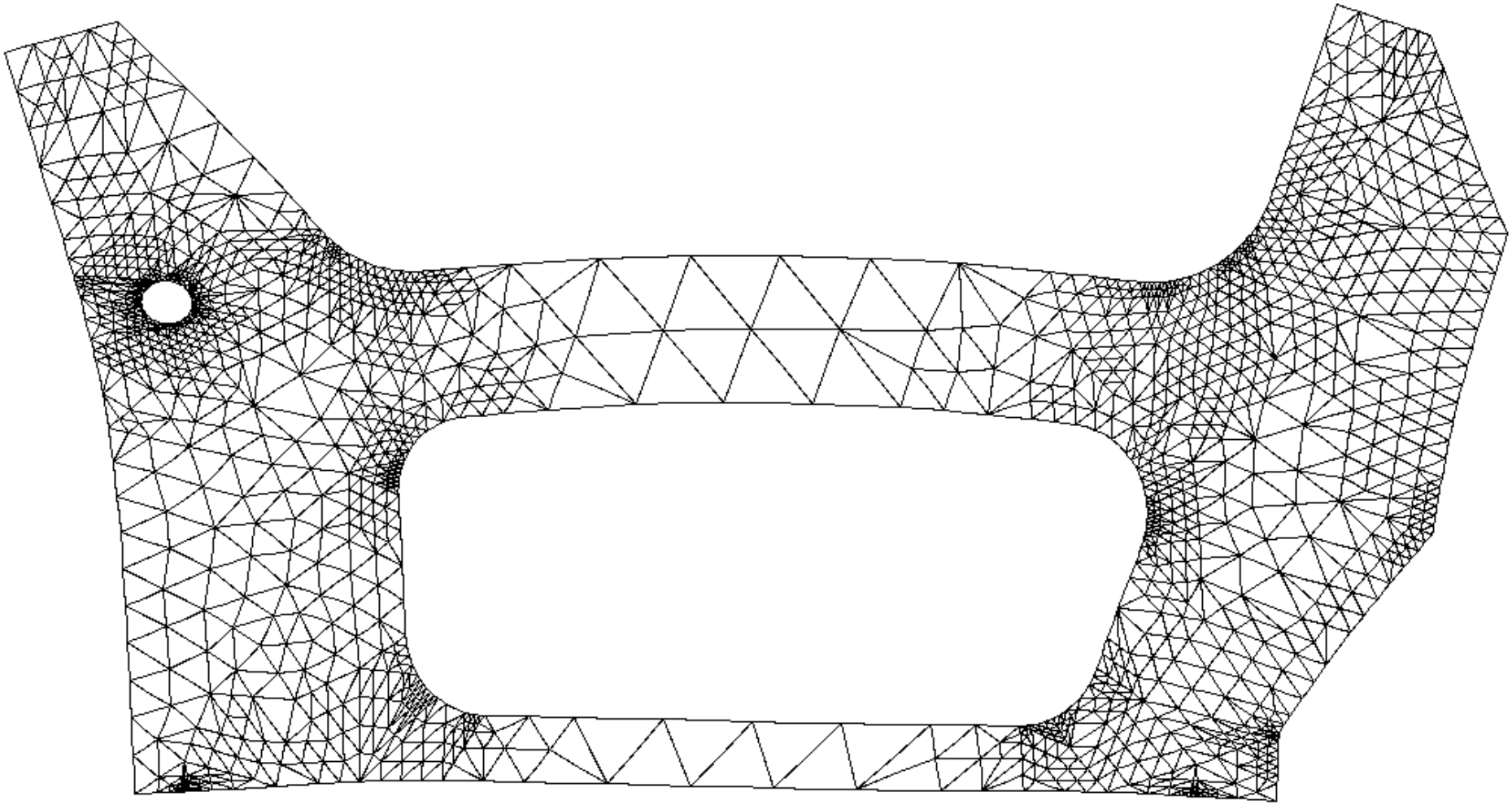


Hole1-12FHG: Schliesskraft 300 t



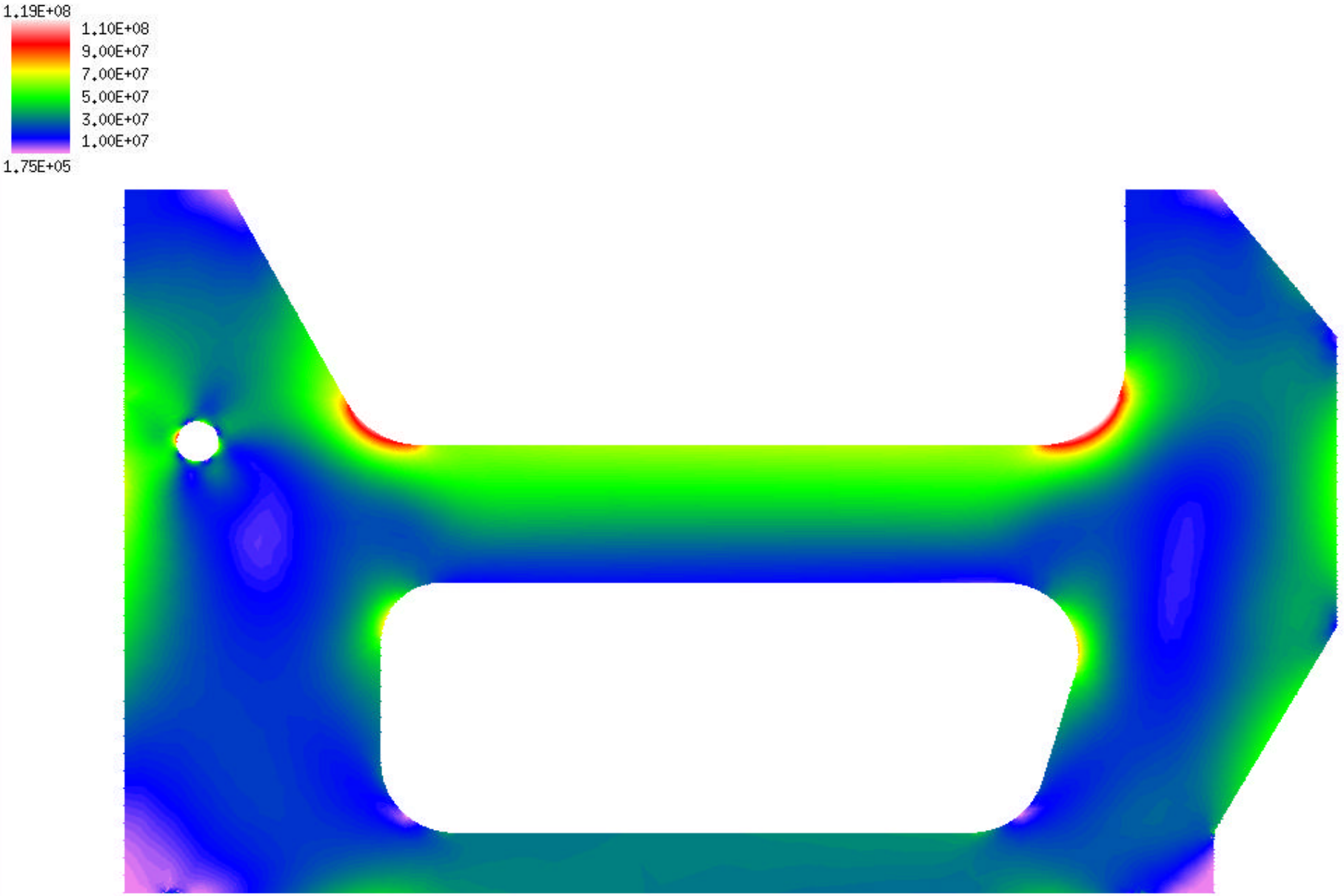
p..pause on/off d..drawing on/off cur..rotate shift+cur..move f1/f2 zoom in/out

Hole1-12FHG: Schliesskraft 300 t



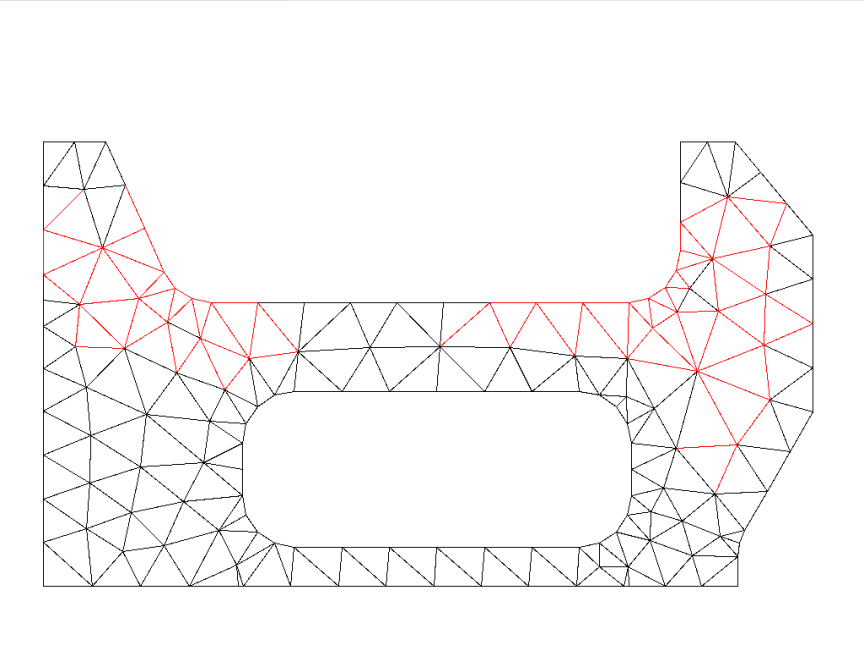
p..pause on/off d..drawing on/off cur..rotate shift+cur..move f1/f2 zoom in/out

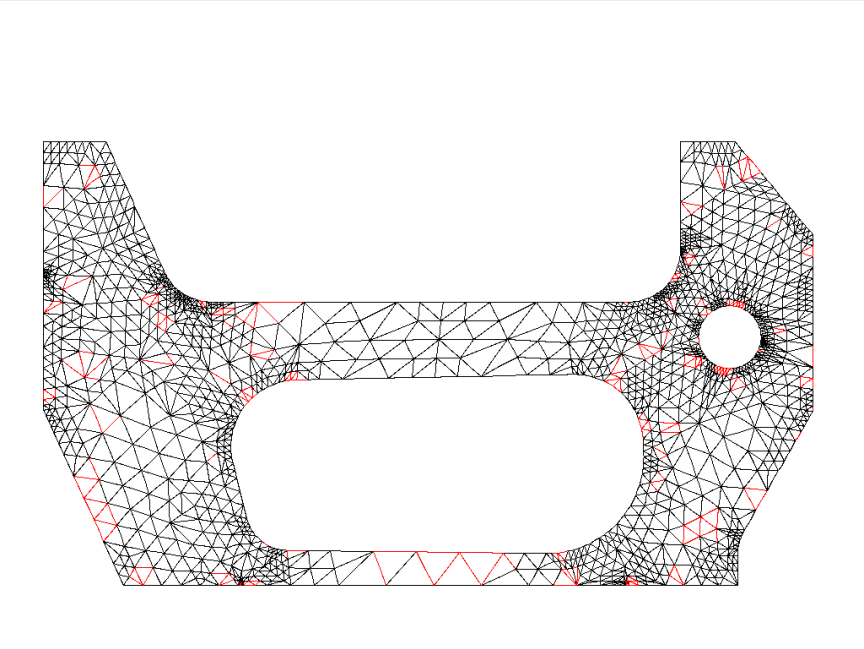
Hole1-12FHG: Schliesskraft 300 t



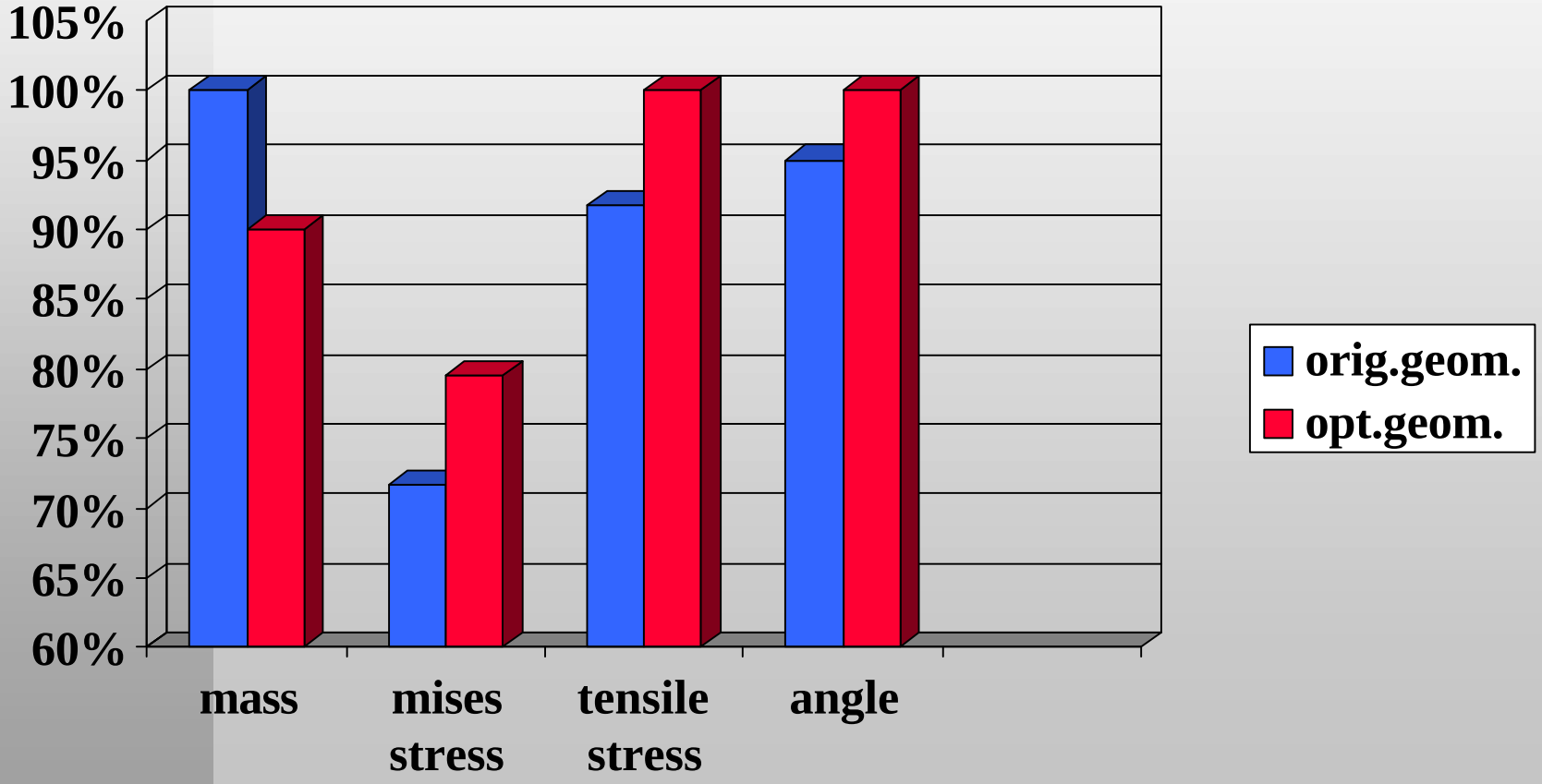
p..pause on/off d..drawing on/off cur..rotate shift+cur..move f1/f2 zc

Original and optimized shape



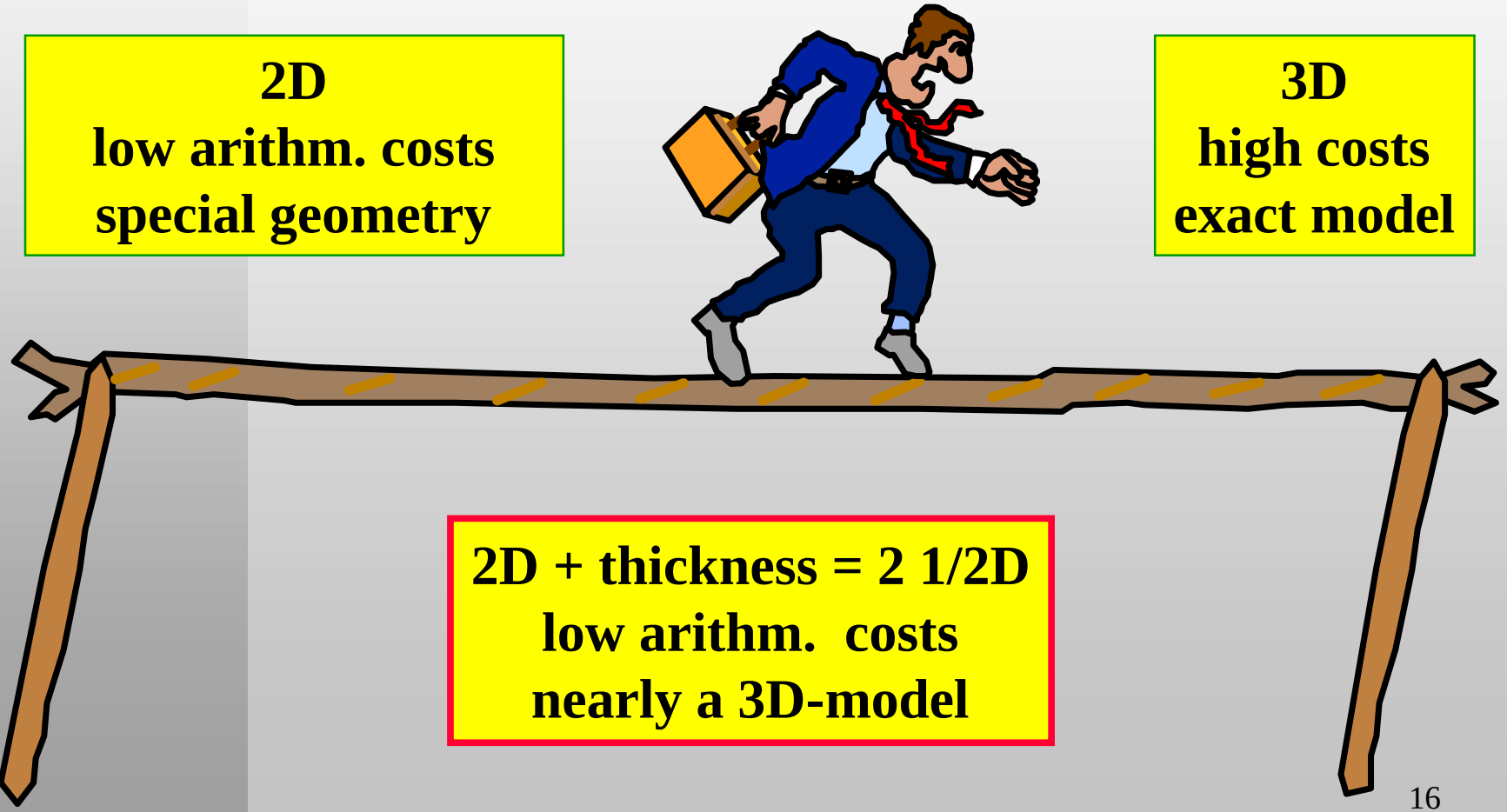


Results of 2D shape optimization



Choice of model for cast iron frames

2D
low arithm. costs
special geometry

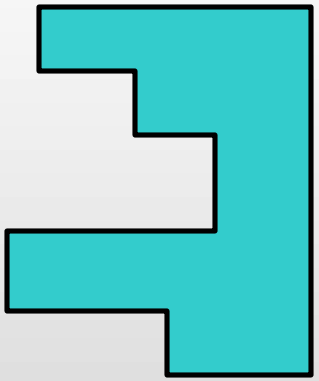


3D
high costs
exact model

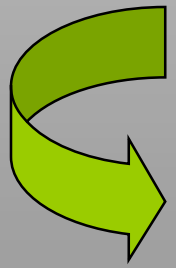
2D + thickness = 2 1/2D
low arithm. costs
nearly a 3D-model

2 1/2D : optimal sizing

- **Sheet metal: discrete thickness optimization**
  - **a few parameters**

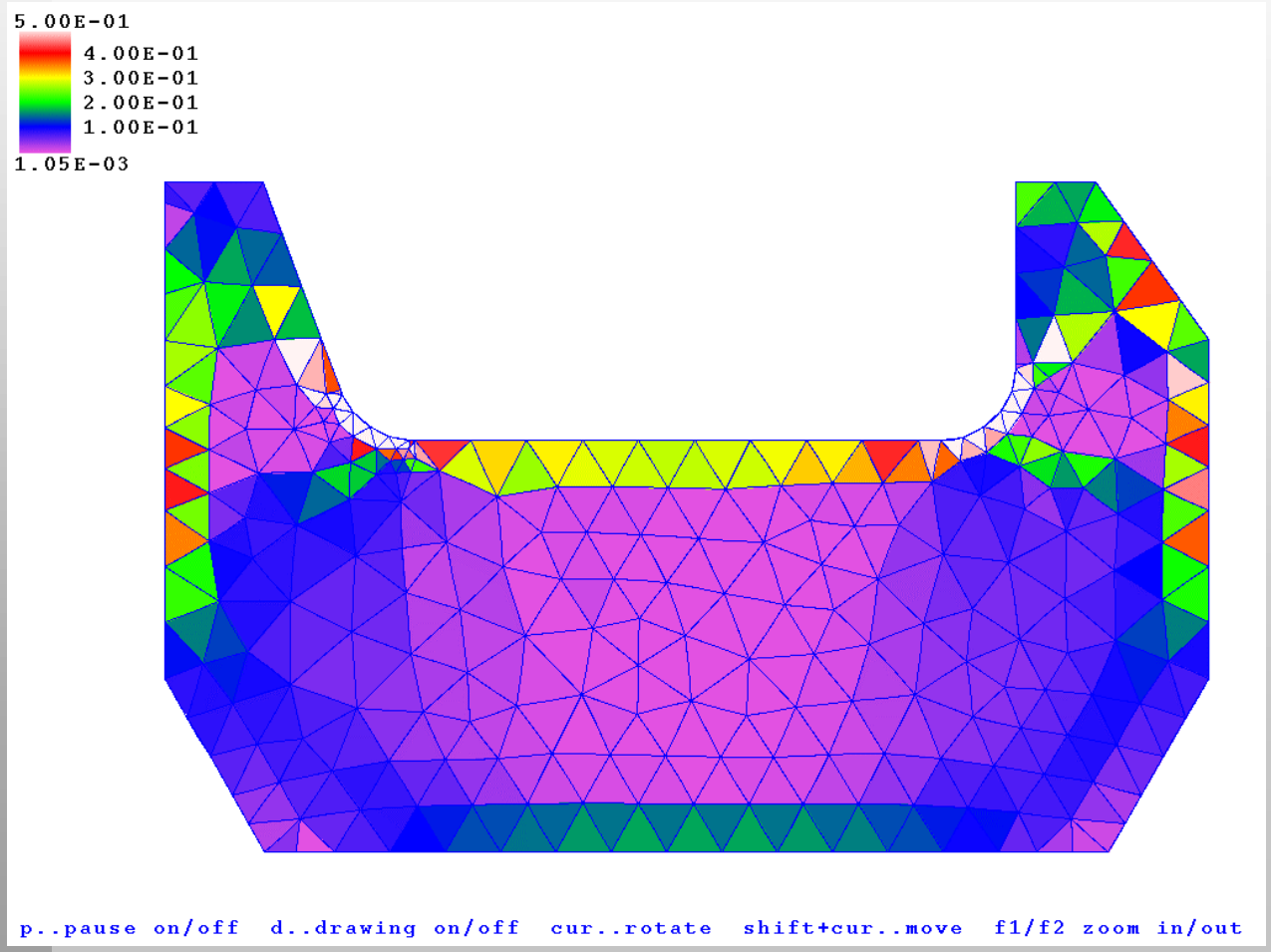


- **Cast iron: continuous thickness optimization**
  - **(arbitrary) many parameters**

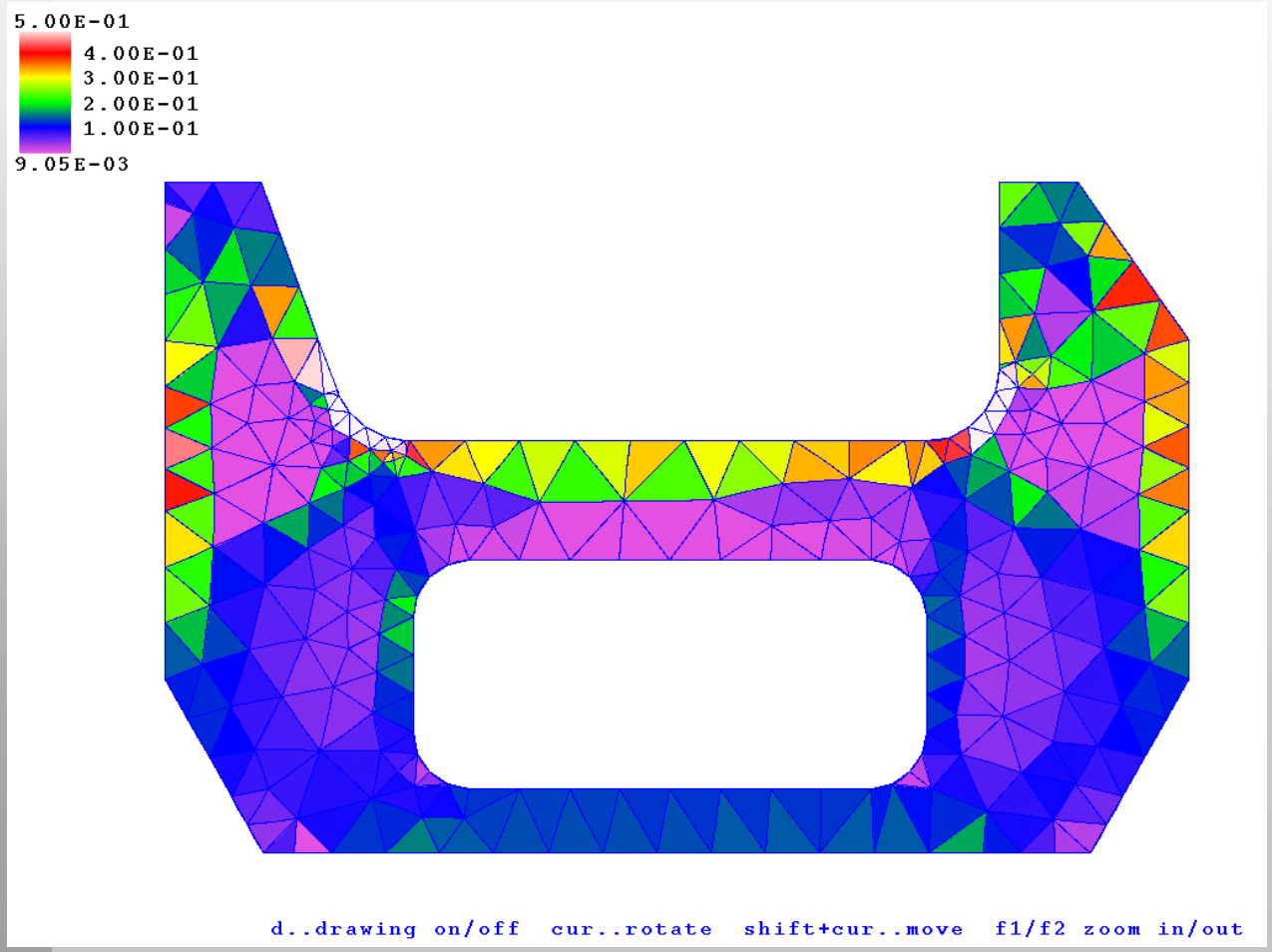


Intelligent Optimization

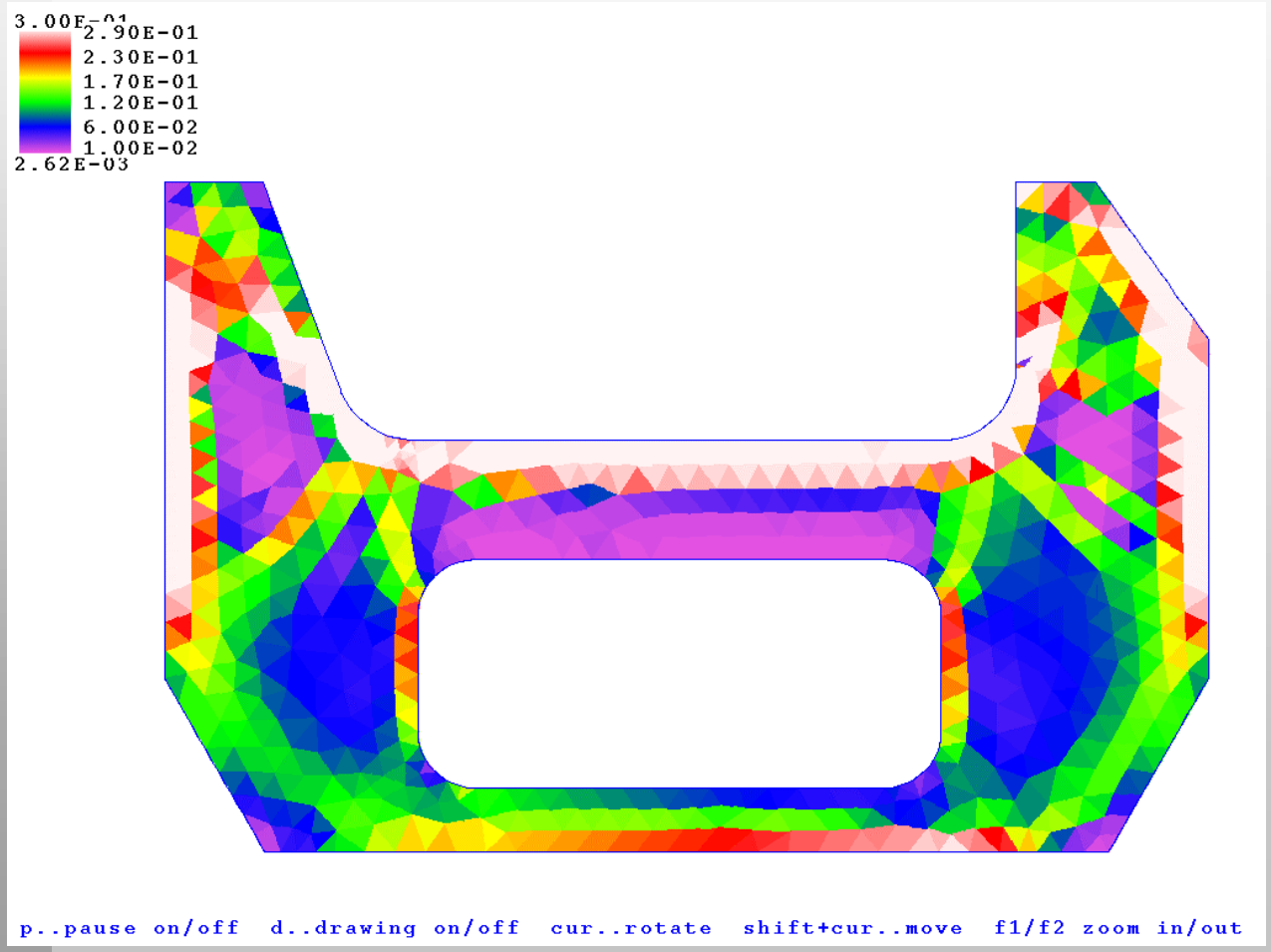
Optimal sizing :no hole



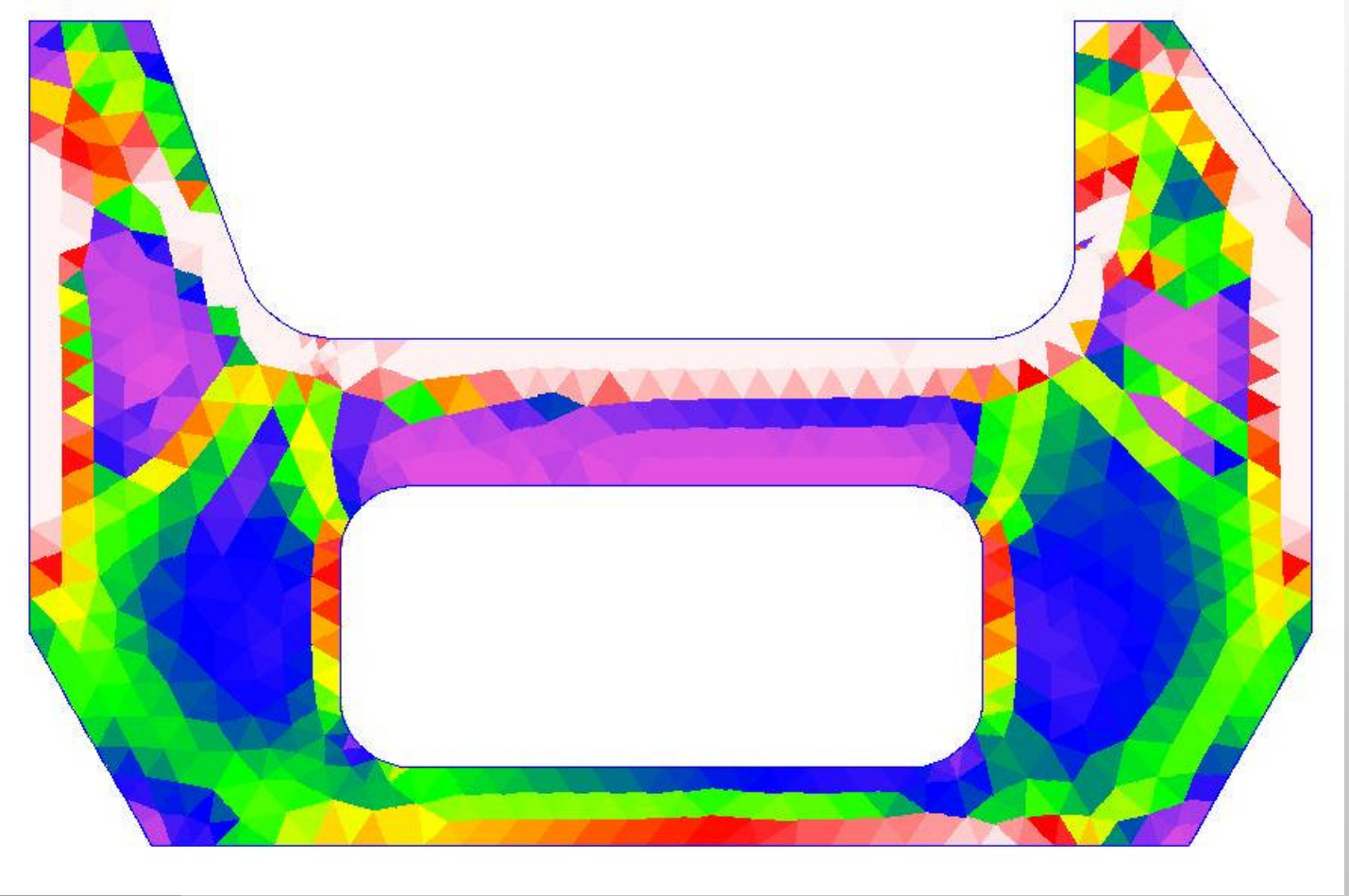
Optimal sizing: 399 parameter



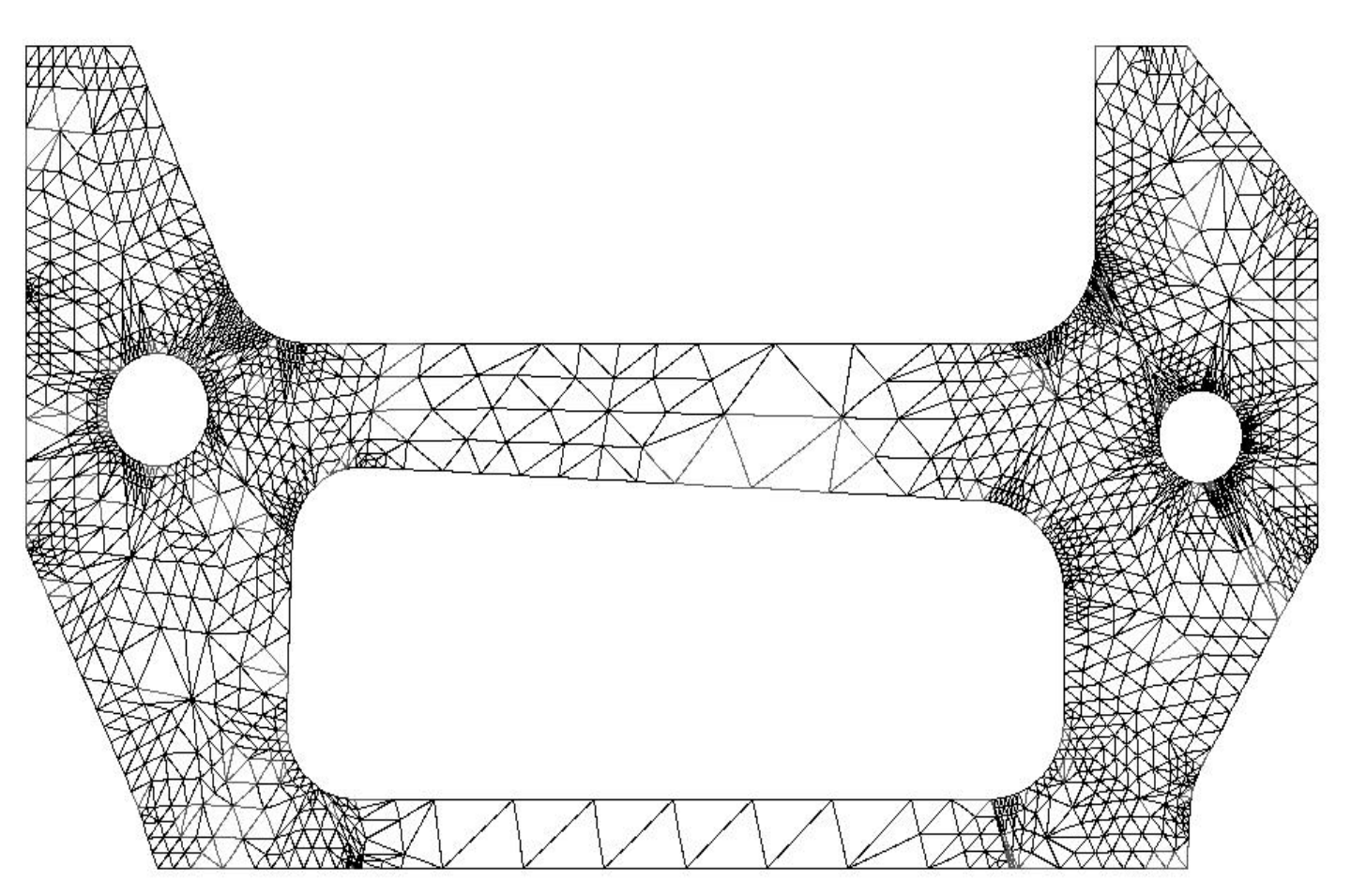
Optimal sizing: 1078 parameter



Optimal sizing



Shape optimization



Results for more design parameters

	Pure AD	Hybrid	Hybrid	Hybrid
# param. **# d.o.f.**	449 7.518	449 7.518	449 29.402	**1078** 9.028
File size (tracing)	**953 MB**	32 MB	129 MB	**78 MB**
#iterations **CPU time**	800 **38,5 h**	800 **3,8 h**	800 14,1 h	2.200 **105,1 h**
T_optimizer **T_gradient** **T_function**	4,0 h **18,0 h** 16,5 h	**1,93 h** **0,54 h** 1,29 h	2,64 h 3,35 h 8,13 h	**90,3 h** 3,9 h 9,8 h

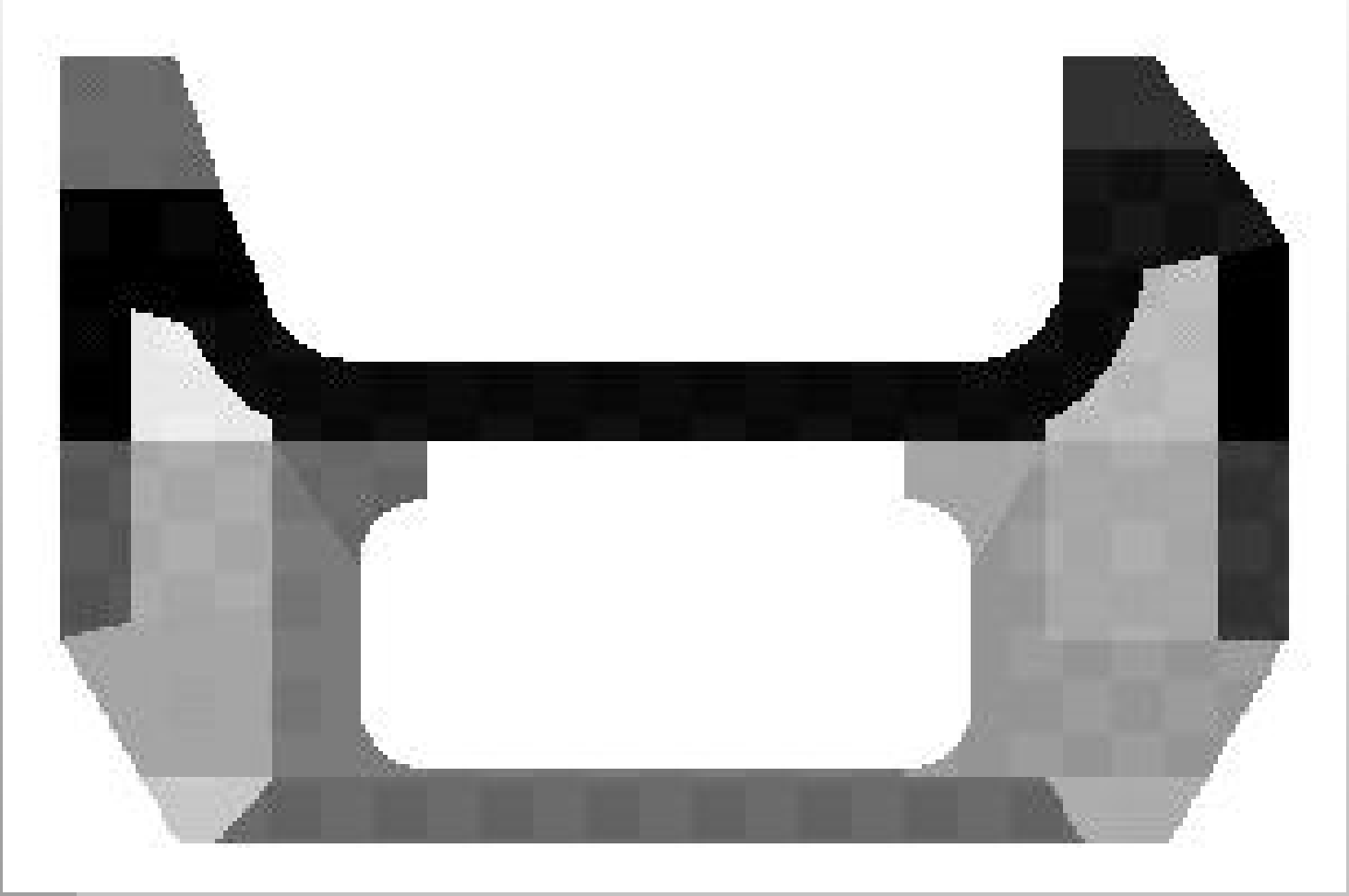
Optimal Sizing: Future Work

- Optimal sizing as **initial guess** for shape optimization
- Reduce #design parameters
- **Use of B-splines** in optimal sizing to reduce number of design variables

Comparison

	Finite Differences	**Adjoint Method**	**Automatic Diff.**
Design parameters	Only a **few**	**Adjustable** to nr. of param.	**many**
Functional	Very **complex**	Rather **simple**	Moderate complex
Flexibility	**high**	**low**	**high**
Iterative solvers	**yes**	**yes**	**no**
CPU-time	Very **high**	**low**	moderate

Optimal sizing :24 parameters



Results for 24 design parameters

	Finite Differences	Pure AD	Hybrid Method
# param.	24	24	24
# d.o.f.	16.690	16.690	16.690
#iterations	83	100	100
#function	**19.752**	315	**236**
CPU time	12,6 h	8,4 h	0,4 h
T_optimizer	0,01h	0,03 h	0,01 h
T_gradient	**12,40 h**	**6,00 h**	**0,18 h**
T_function	0,23 h	2,36 h	0,20 h

Conclusions – part I

- **Flexible** method for optimal sizing
- Objectives and constraints of **moderate complexity**
- **Iterative solvers** for state equation
- Huge evaluation graphs are avoided
- Optimal sizing as **initial guess** for shape optimization
- **Use of B-splines** in optimal sizing to reduce number of design variables

Mathematical description

- **Minimize a functional f**

$$f(\mathrm{u}(v_D), v_D) \rightarrow \text{minimum}$$

$$\nabla f = 0$$

- **under the given constraints**
- **solving as direct problem the plane stress state**

$$-\mu\Delta\,\mathrm{u} - (\lambda+\mu)\nabla\,\mathrm{div}\,\mathrm{u} = \mathrm{b}$$

Constraints

- **Besides an admissible geometry $\Omega(v_D)$**

$$\sigma(x) \le \sigma_{max} \qquad \text{mises stress}$$

$$\alpha(x) \le \alpha_{max} \qquad \text{shrinking angle}$$

$$\tau(x) \le \tau_{max} \qquad \text{tensile stress}$$

$$\forall x \in \Omega(v_D)$$

- **and ????**

Functional

$$f(v_D, \mathrm{u}, \sigma, \alpha) = \omega_m \cdot \mathrm{mass}(v_D) / m_0$$
$$= \omega_\sigma \cdot 10^4 \left(\sum \left[\max(\sigma(x+\mathrm{u})/\sigma_{\max} \text{ -1.0, 0}) \right]^2 \right)$$
$$= \omega_{\alpha,1} \cdot 10^4 \left(\left[\max(\alpha(x+\mathrm{u})/\alpha_{\max} \text{ -1.0, 0}) \right]^2 \right)$$
$$= \omega_{\alpha,2} \cdot \alpha(x+\mathrm{u})/\alpha_{soft}$$

- **Barrier functions** $\in C^1$
- **weights** ω_{xx} **: criteria for engineer**
- **below barrier of angle:** $\alpha(x+\mathrm{u})/\alpha_{soft}$

Optimization algorithm

- **Nonlinear box constraints**
- **continuous subproblems**

$$\nabla f = 0$$

- **Quasinewton method using**

$$\upsilon_D \leftarrow \upsilon_D - H_f^{-1} \cdot \nabla f(\upsilon_D)$$

Gradient of functional

- **When minimizing a functional f**

$$f(\underline{u}(\underline{v}_D, \Omega(\underline{v}_D)), \Omega(\underline{v}_D)) \rightarrow \min.$$

- **we need derivative**

$$\frac{df}{d\Omega} = \frac{\partial f}{\partial \Omega} + \frac{\partial f}{\partial \underline{u}} \frac{\partial \underline{u}}{\partial \Omega}$$

- **and finally**

$$\frac{df}{dv_i} = \frac{\partial f}{\partial v_i} + \frac{\partial f}{\partial \underline{u}} \frac{\partial \underline{u}}{\partial v_i} \qquad i = 1, n_D$$

$\frac{df(v_D)}{dv_i}$ by finite differences

$$\frac{df(v_D)}{dv_i} = \lim_{\tau \to 0^+} \frac{1}{\tau}\left[\underline{f}(\underline{v}_D + \tau \underline{e}_i) - f(\underline{v}_D)\right] \qquad i = 1, n_D$$

- **with**

$$K(\underline{v}_D + \tau \underline{e}_i) \cdot \underline{u}(\underline{v}_D + \tau \underline{e}_i) = \underline{b}(\underline{v}_D + \tau \underline{e}_i) \qquad i = 1, n_D$$

- **requires** $> n_D + 1$ **solves of direct problem per step in optimization**

Direct method

$$K \cdot \underline{u} = \underline{b} \xrightarrow{\frac{\partial}{\partial v_i}} \frac{\partial K}{\partial v_i}\underline{u} + K\frac{\partial \underline{u}}{\partial v_i} = \frac{\partial \underline{b}}{\partial v_i}$$

$$\Rightarrow \quad \frac{\partial \underline{u}}{\partial v_i} = K^{-1} \cdot \left(\frac{\partial \underline{b}}{\partial v_i} - \frac{\partial K}{\partial v_i}\underline{u} \right)$$

- **results in derivatives of functional**

$$\frac{df}{dv_i} = \frac{\partial f}{\partial v_i} + \frac{\partial f}{\partial \underline{u}} \cdot K^{-1} \cdot \left(\frac{\partial \underline{b}}{\partial v_i} - \frac{\partial K}{\partial v_i}\underline{u} \right) \qquad i = 1, n_D$$

- **still** n_D **direct solves needed**

Adjoint method I

- **The substitution**

$$\frac{\partial f}{\partial \underline{u}} \cdot K^{-1} = \left(\left[K^{-1} \right]^T \cdot \left[\frac{\partial f}{\partial \underline{u}} \right]^T \right)^T = \underline{\lambda}^T$$

- **requires solving of**

- **1 adjoint problem**

$$K^T \cdot \underline{\lambda} = \left[\frac{\partial f}{\partial \underline{u}} \right]^T$$

Adjoint method II

- **Therefore**

$$\frac{df}{dv_i} = \frac{\partial f}{\partial v_i} + \underline{\lambda}^T \cdot \left(\frac{\partial \underline{b}}{\partial v_i} - \frac{\partial K}{\partial v_i} \underline{u} \right) \qquad i = 1, n_D$$

- **requires solution of only 1 direct problem**

$$K^T \cdot \underline{\lambda} = \left[\frac{\partial f}{\partial \underline{u}} \right]^T$$

Adjoint method III

- **Matrix derivatives**

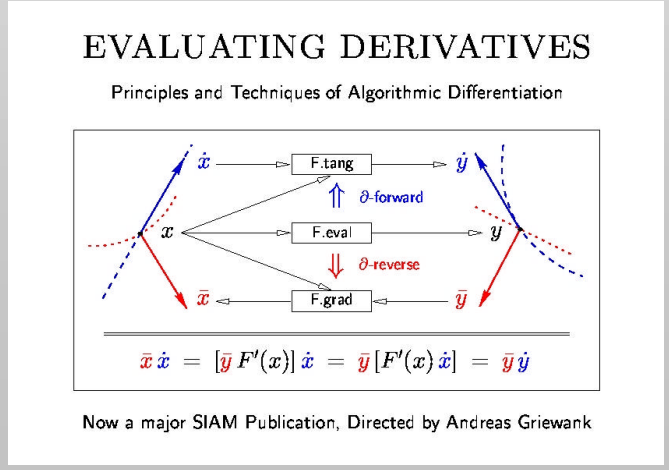
$$\frac{\partial K(\underline{v}_D, x(\underline{v}_D))}{\partial v_i} = \frac{\partial K}{\partial v_i} + \frac{\partial K}{\partial x} \cdot \frac{\partial x}{\partial v_i} \qquad i = 1, n_D$$

- **Optimal sizing:** $\frac{\partial K}{\partial v_i}$ **easy,** $\frac{\partial K}{\partial s}, \frac{\partial s}{\partial v_i} \neq 0$
- **Shape optimization:** $\frac{\partial K}{\partial v_i} = 0$

  coding for $\frac{\partial K}{\partial x}, \frac{\partial x}{\partial v_i}$

Automatic Differentiation

- **Calculate** $\frac{df}{dv_i}$ **via a routine derived from the code for calculating** f
- **ADOL-C [A. Griewank, Dresden] (runtime generation of evaluation graph)**



Hybrid Method (AD + adjoint)

$$\frac{df}{dv_i} = \frac{\partial f}{\partial v_i} + \frac{\partial f}{\partial \underline{u}} \cdot K^{-1} \cdot \left(\frac{\partial \underline{b}}{\partial v_i} - \frac{\partial K}{\partial v_i} \underline{u} \right)$$

Automatic differentiation: $\frac{\partial f}{\partial v_i}, \frac{\partial f}{\partial \underline{u}}$

Coded in subroutines: $\frac{\partial \underline{b}}{\partial v_i}, \frac{\partial K}{\partial v_i}$ $\frac{\partial K}{\partial x}, \frac{\partial x}{\partial v_i}$

Shape Optimization - Gradient

Design functional evaluation means:

1) Take a **set of designparameters**

2) Generate the **geometrical shape**

3) Create a **mesh** for the current shape

4) Calculate the **solution** on the current mesh

5) **Evaluate the objective**

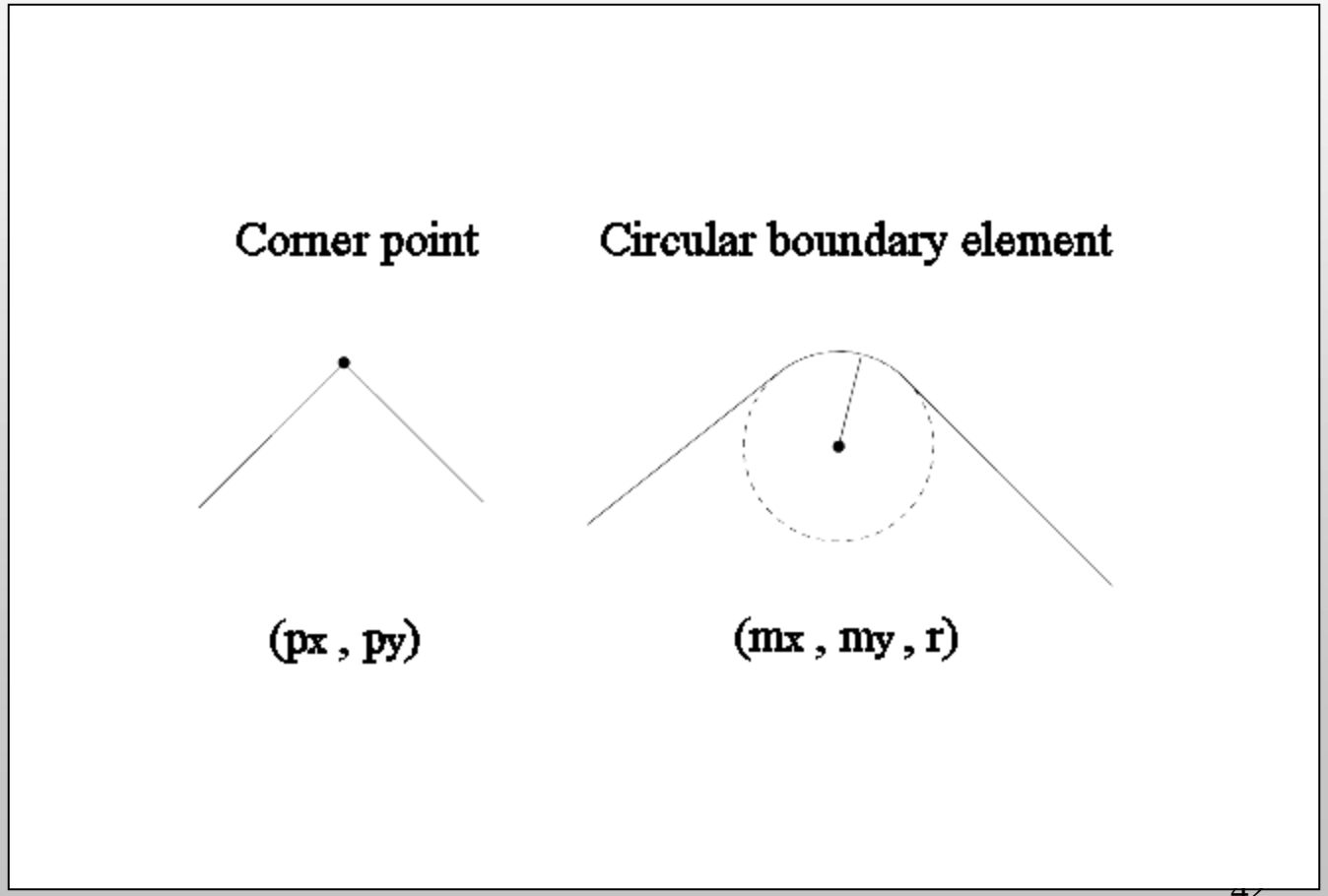
Evaluating the functional

1) Parameters

2) Shape

3) Mesh

4) Solution



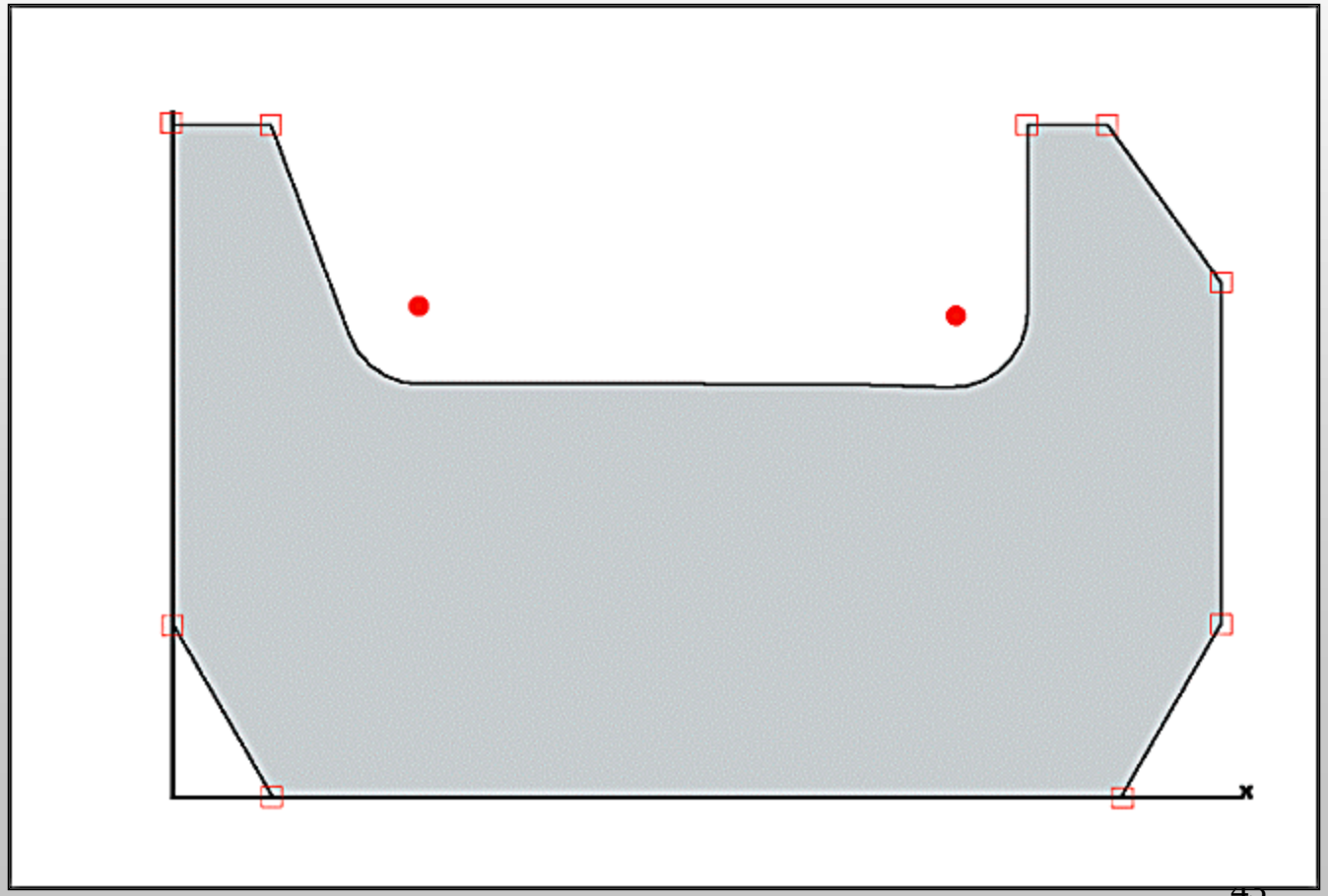
Evaluating the functional

1) Parameters

2) Shape

3) Mesh

4) Solution



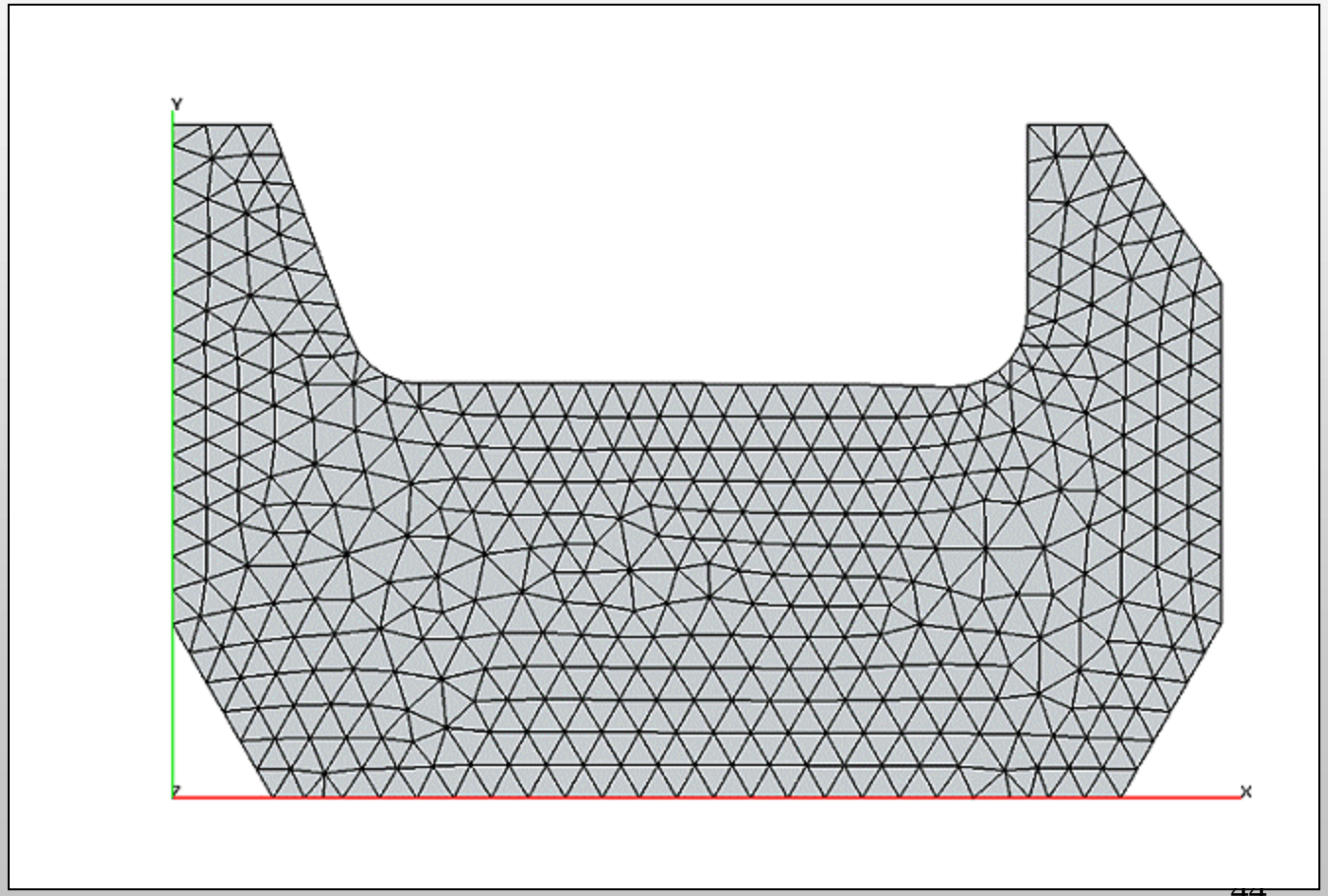
Evaluating the functional

1) Parameters

2) Shape

3) Mesh

4) Solution



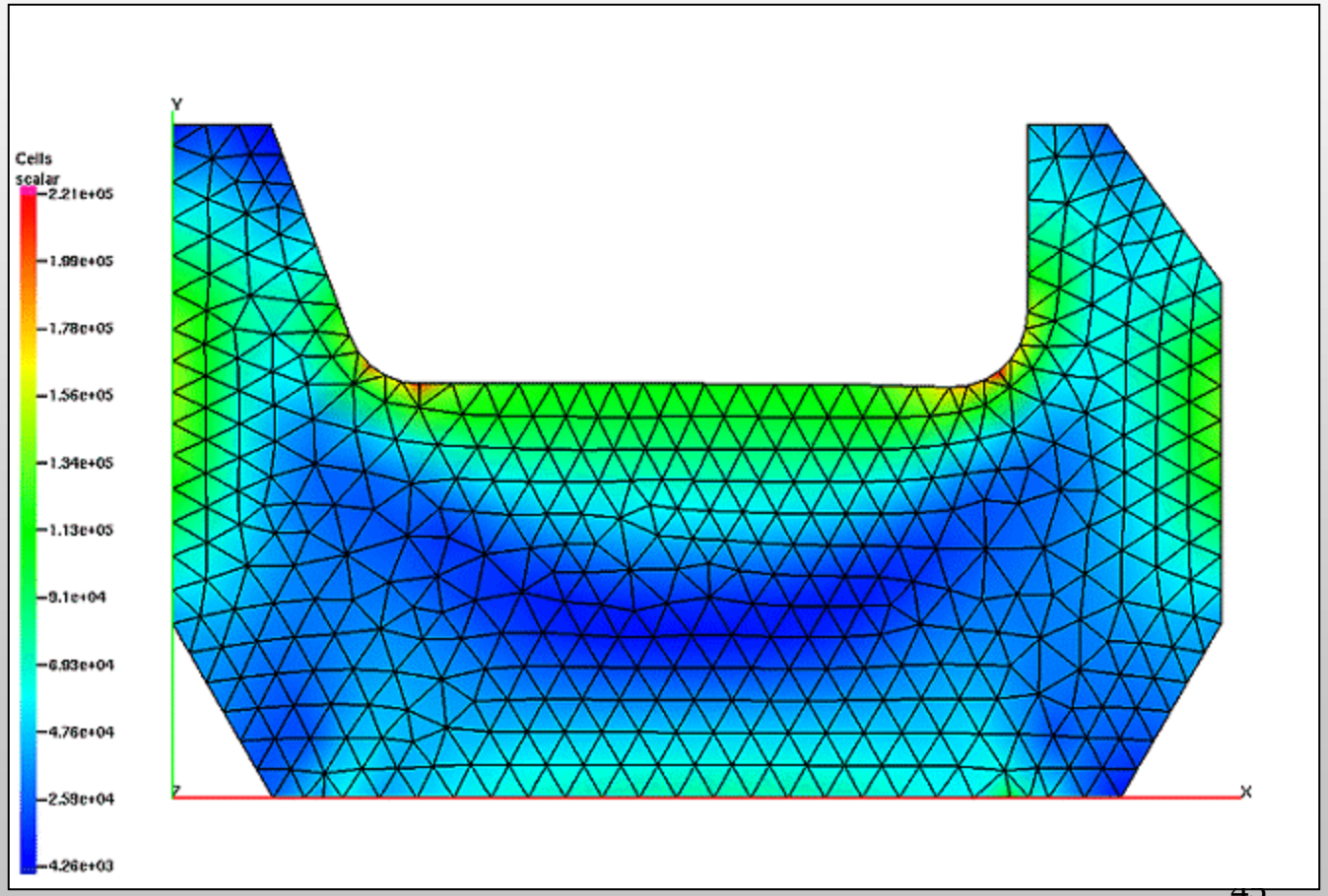
Evaluating the functional

1) Parameters

2) Shape

3) Mesh

4) Solution



More about derivatives

$$\frac{df}{dv_i} = \frac{\partial f}{\partial v_i} + \frac{\partial f}{\partial \underline{u}} \cdot K^{-1} \cdot \left(\frac{\partial \underline{b}}{\partial v_i} - \frac{\partial K}{\partial v_i} \underline{u} \right)$$

$$\frac{\partial f}{\partial \underline{u}}$$

The functional depends on the **displacements**...
...for example by angular constraints...

$$\frac{\partial f}{\partial v_i}, \frac{\partial \underline{b}}{\partial v_i}, \frac{\partial K}{\partial v_i}$$

...but there are also dependencies on the **design parameters**.

Calculating the derivative

So what we really get when calculating the derivative of K looks actually like that (written in a very crude, but intuitive way...):

$$\frac{\partial K}{\partial parameter} = \frac{\partial K}{\partial mesh} \cdot \frac{\partial mesh}{\partial boundary} \cdot \frac{\partial boundary}{\partial parameter}$$

Though mathematically simple, this gets rather complicated to implement (eg. Changing a radius changes two tangential points and therefore the two adjacient straight lines)

Calculating the derivative

So what we really get when calculating the derivative of K looks actually like that (written in a very crude, but intuitive way...):

$$\frac{\partial K}{\partial parameter} = \frac{\partial K}{\partial mesh} \cdot \frac{\partial mesh}{\partial surface} \cdot \frac{\partial surface}{\partial parameter}$$

This means differentiating the solution of the linear elastic subproblem by the dirichlet boundary values. In a direct approach this requires the solution of one field problem per parameter.

Calculating the derivative

So what we really get when calculating the derivative of K looks actually like that (written in a very crude, but intuitive way...):

$$\frac{\partial K}{\partial parameter} = \frac{\partial K}{\partial mesh} \cdot \frac{\partial mesh}{\partial surface} \cdot \frac{\partial surface}{\partial parameter}$$

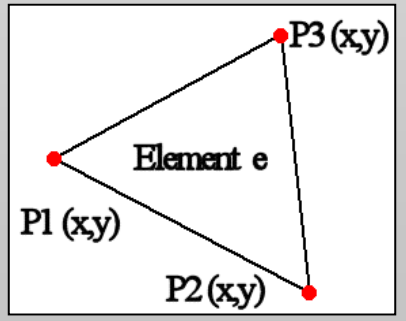
Finally this can be reduced to differentiating the element matrices with respect to their respective cornerpoints, which is fairly uncomplicated.

A closer look at K

The Stiffness Matrix K is usually calculated as the sum of all **Element Stiffness Matrices**.

$$K = \sum_{Elements} C_e^{\,T} \cdot K_e \cdot C_e$$

These, of course, depend on the **corner points** of their respective element.



As these are part of the mesh, **K depends on the mesh**.

The mesh mapping I

The mesh naturally depends on the outer shape.

But generating a new mesh for each outer shape leads to serious problems:

- Similar shapes may lead to locally completely different meshes, the mapping is not continuous, and therefore of course not differentiable.

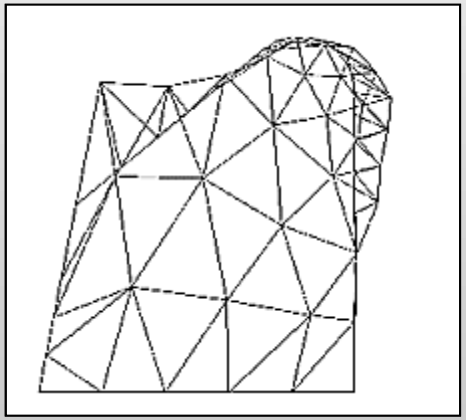
- Mesh generation is time consuming.

The mesh mapping II

A better idea is to deform the mesh in a way that it fits into the new outer shape. A first approach takes two steps:

1) Map surface nodes to the new outer shape

2) Use smoothing algorithm to improve mesh (eg.Jacobi)

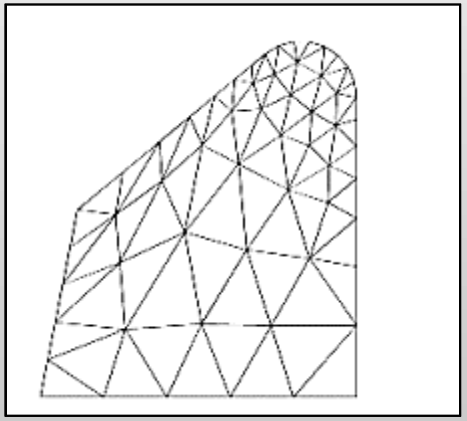


The mesh mapping II

A better idea is to deform the mesh in a way that it fits into the new outer shape. A first approach takes two steps:

1) Map surface nodes to the new outer shape

2) Use smoothing algorithm to improve mesh (eg.Jacobi)



Problem: The mapping is differentiable, but it takes many smoothing steps to obtain a reasonable mesh. Hard to get rid of overlapping elements!

The mesh mapping III

Finally a good (and also very intuitive) approach is to solve an elastic subproblem for the mesh.

This strategy is:

Fast

It is possible to use highly advanced solution strategies for the linear FEM system involved. The computation time no longer depends on the degree of deformation (as was the case for the Jacobi smoother)

The mesh mapping III

Finally a good (and also very intuitive) approach is to solve an elastic subproblem for the mesh.

This strategy is:

Fast - Stable

The quality of the resulting mesh is usually very good. This has a direct influence on the stability and speed of the FEM calculations needed for the objective (as they are calculated on this mesh).

The mesh mapping III

Finally a good (and also very intuitive) approach is to solve an elastic subproblem for the mesh.

This strategy is:

Fast - Stable - Flexible

There are a number of rather easy approaches to further improve the quality of the resulting mesh (eg. increasing the stiffness of small mesh elements and thus keeping them from overlapping). The parameters of the elastic problem offer a variety of possibilities.

The mesh mapping III

Finally a good (and also very intuitive) approach is to solve an elastic subproblem for the mesh.

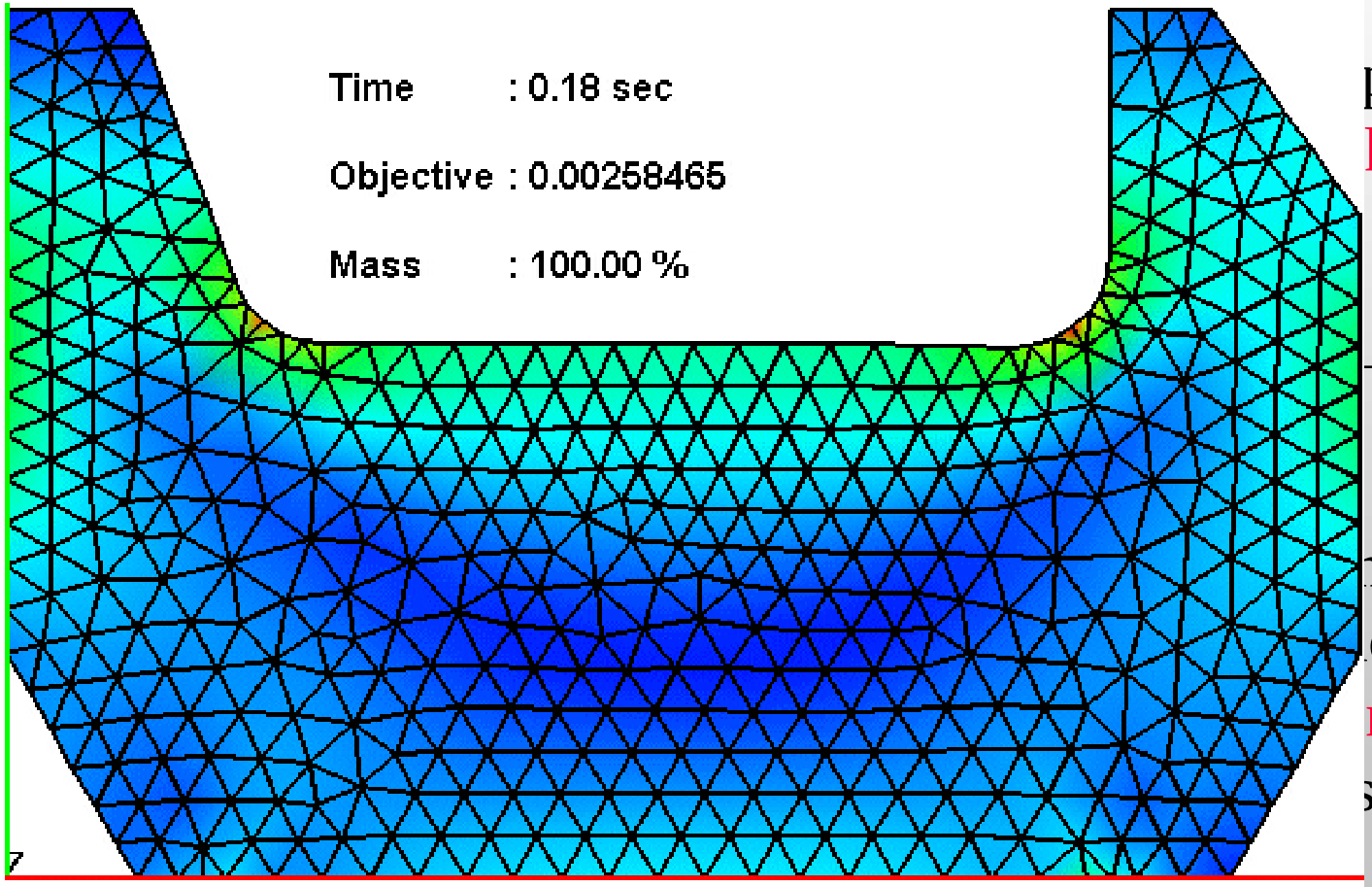
This strategy is:

Movie !!

e - Differentiable

nent field) of the linear elastic
entiably on the dirichlet
herefore the mapping from outer shape to resulting mesh is differentiable as well.

The mesh mapping III

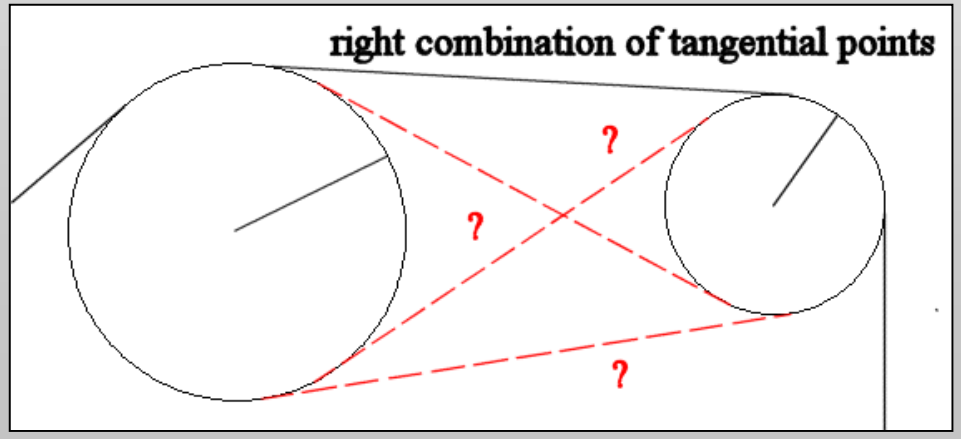


proach is
h.

near elastic
chlet
ng from outer
s well.

The shape mapping

The first thing one has to do is to map the design parameters to the outer shape of the geometry. The mathematical complexity of this is (for the simple approach using only straight lines and circular boundary elements) low, but the implementation can become very tricky.



Summary: Shape Optimizaion

Mesh deformation and especially mesh deformation with linear elasticity algorithms highly improve speed and stability of shape optimization.

This will be especially important for future full 3D applications (enormous increase of elements).

Furthermore it would be useful to implement a more general approach to shape description (splines...)

Optimal Sizing – smooth thickness

- **Matrix derivatives**

$$\frac{\partial K(\underline{v}_D, s(\underline{v}_D))}{\partial v_i} = \frac{\partial K}{\partial v_i} + \frac{\partial K}{\partial s} \cdot \frac{\partial s}{\partial v_i} = \frac{\partial K}{\partial s} \cdot \frac{\partial s}{\partial v_i} \qquad i = 1, n_D$$

$$\frac{\partial K}{\partial v_i} = 0$$

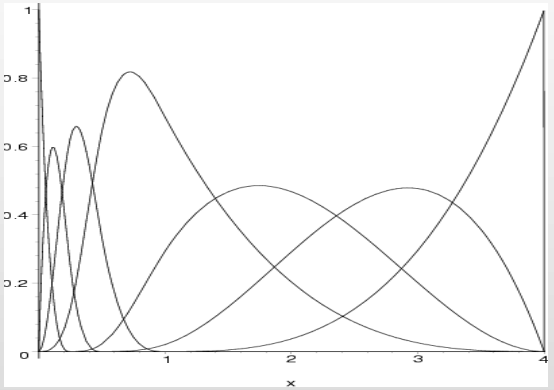
$$\frac{\partial K}{\partial s}, \frac{\partial s}{\partial v_i} \neq 0$$

- **Thickness distribution function**

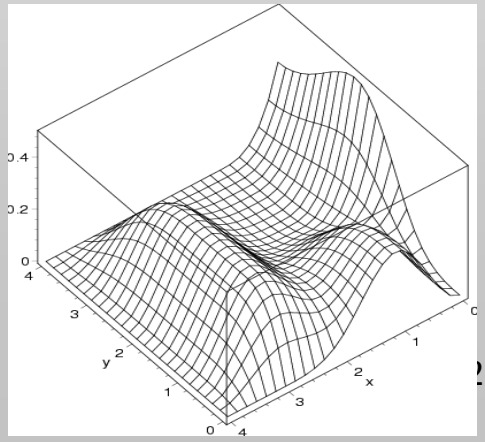
$$s(x) = s(\underline{v}_D(x))$$

B-Spline surface

- B-splines
  - Used in CAD-systems
  - Tensor product mesh
  - Local (cubic) Bezier-functions

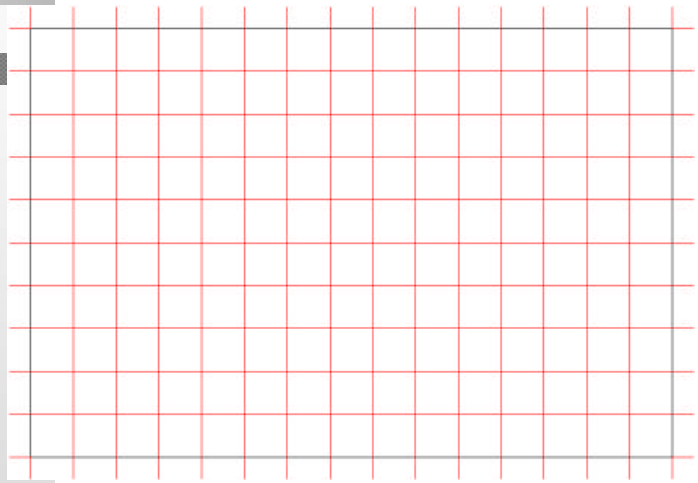


- Tensor product to form 2D-surface
  - Smooth 2D-surface
  - Rectangular domain

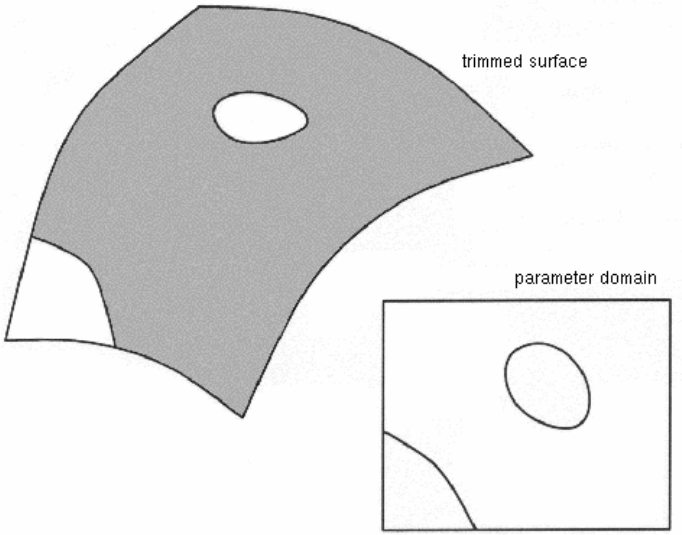


B-Spline surface II

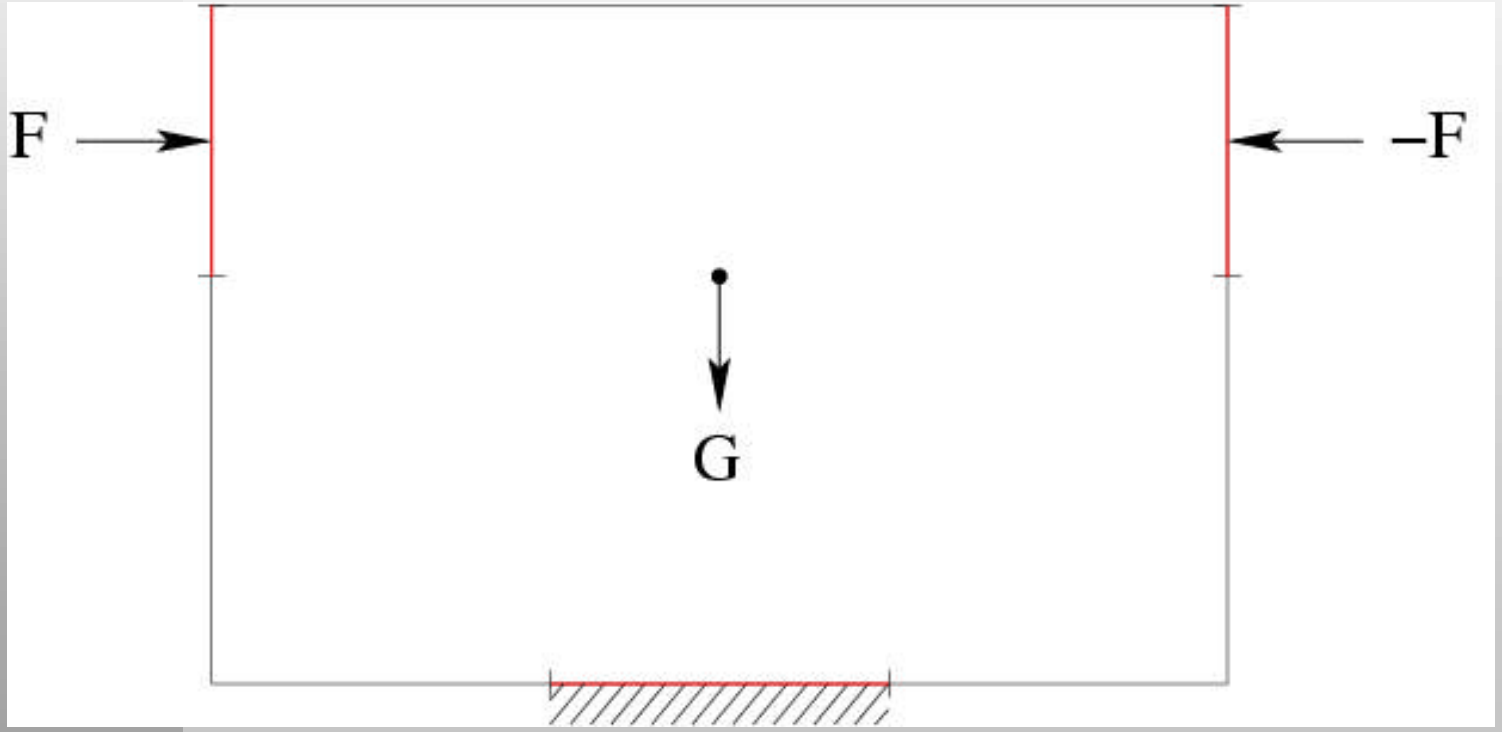
- Tensor product
  - Non-trivial domain??



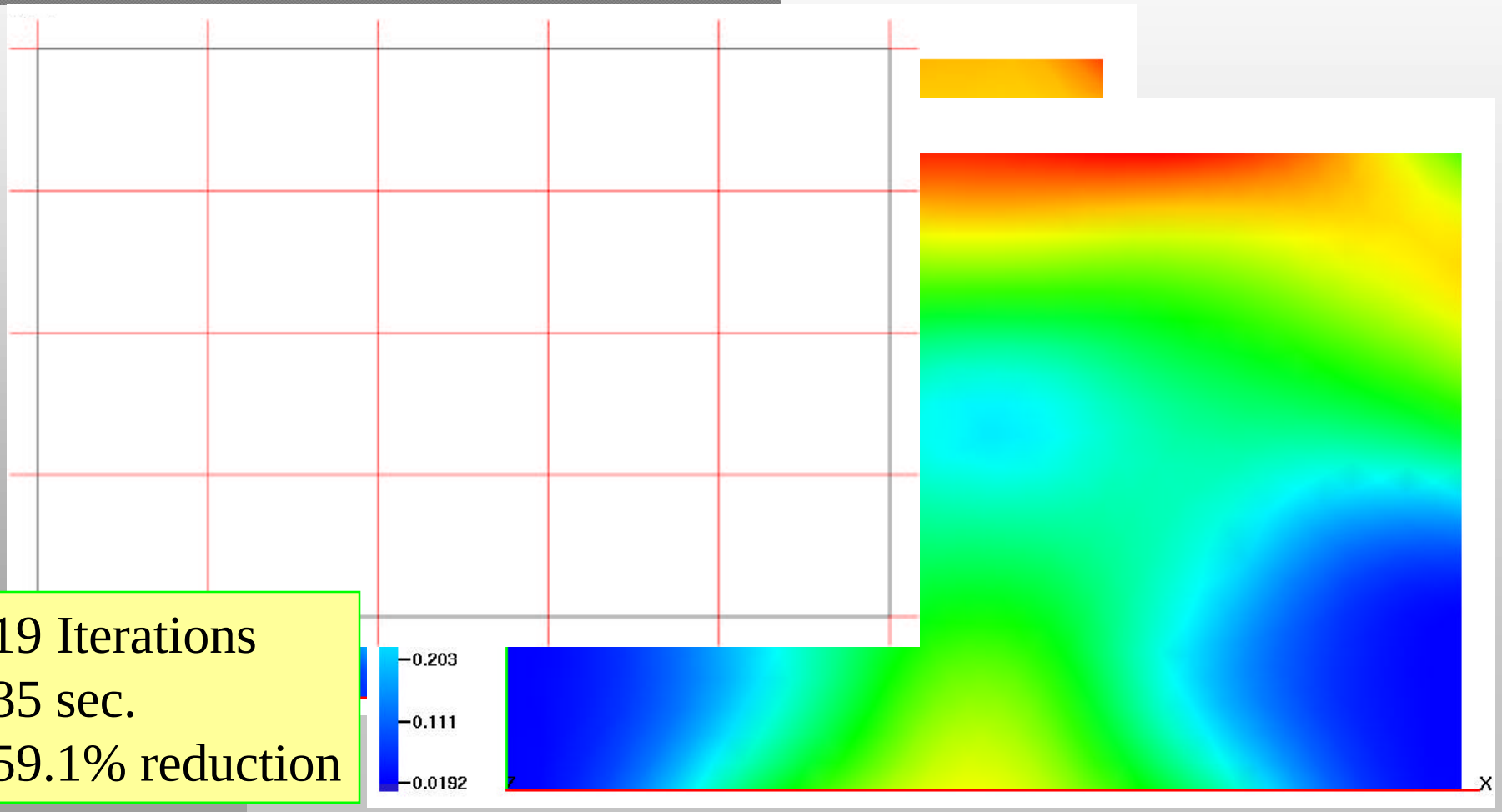
- Trigger surface
  - More Bezier-coeff.
  - Than design variables



B-Spline rectangle

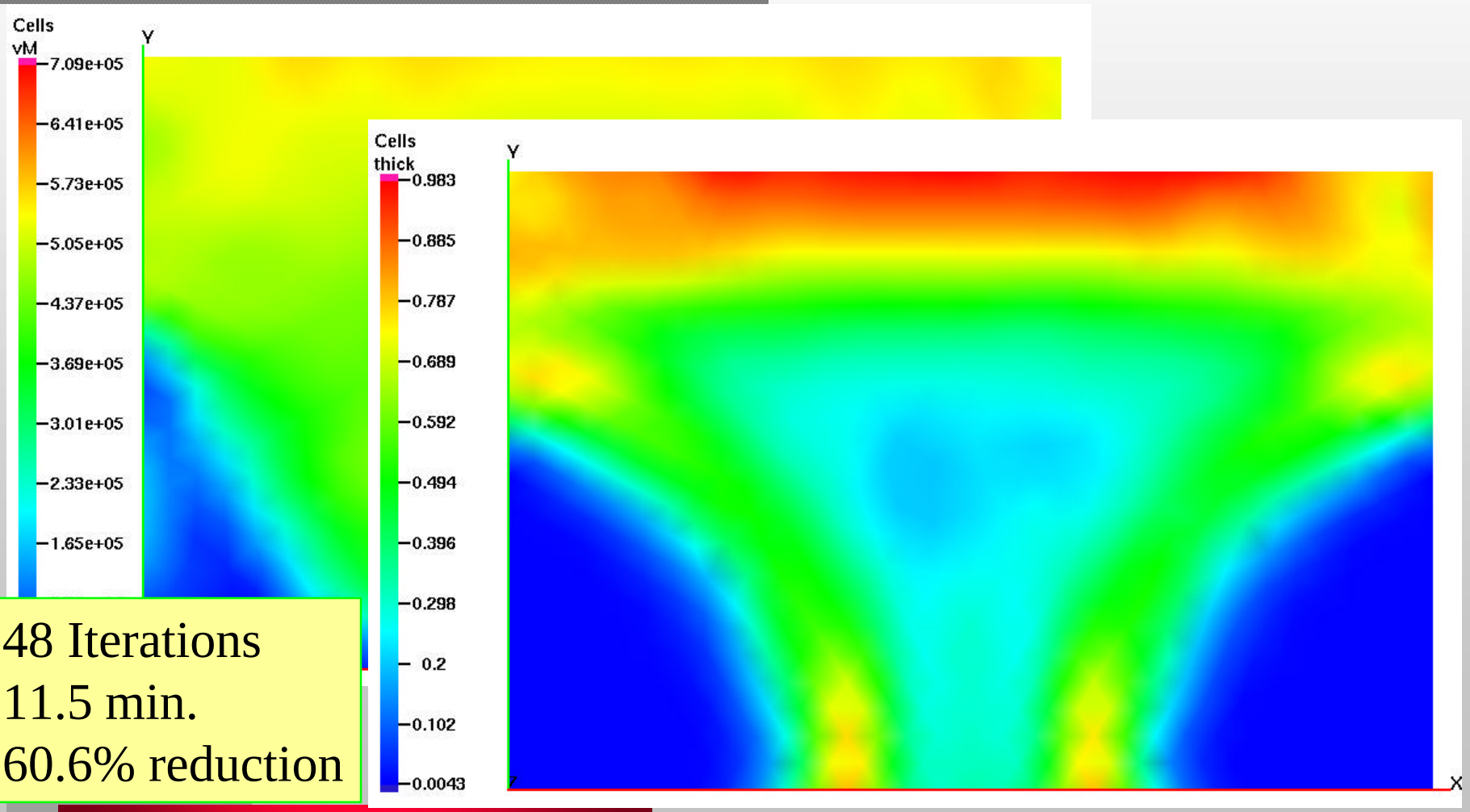


B-Spline rectangle (42 d.v.)



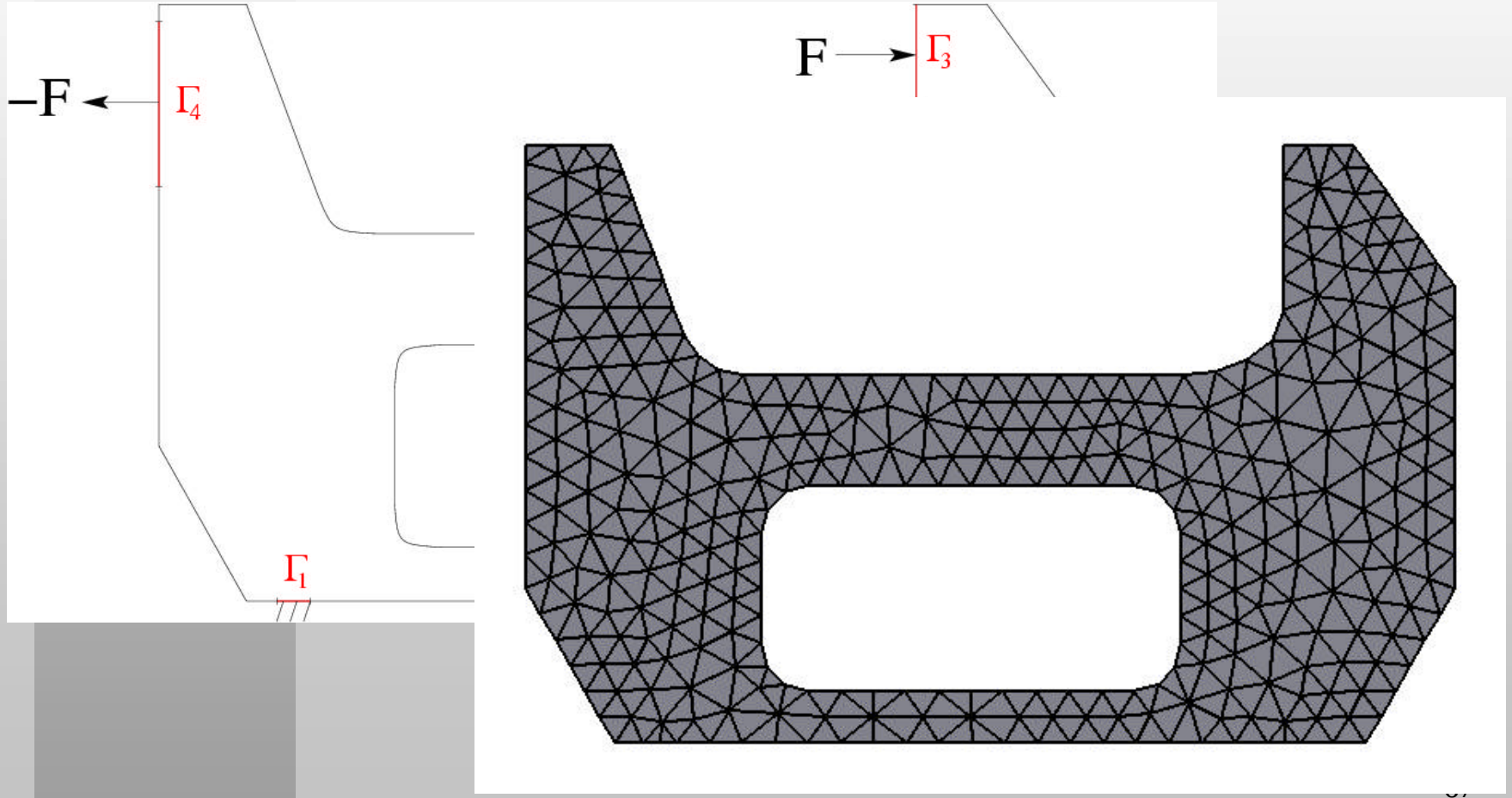
- 19 Iterations
- 35 sec.
- 59.1% reduction

B-Spline rectangle (204 d.v.)

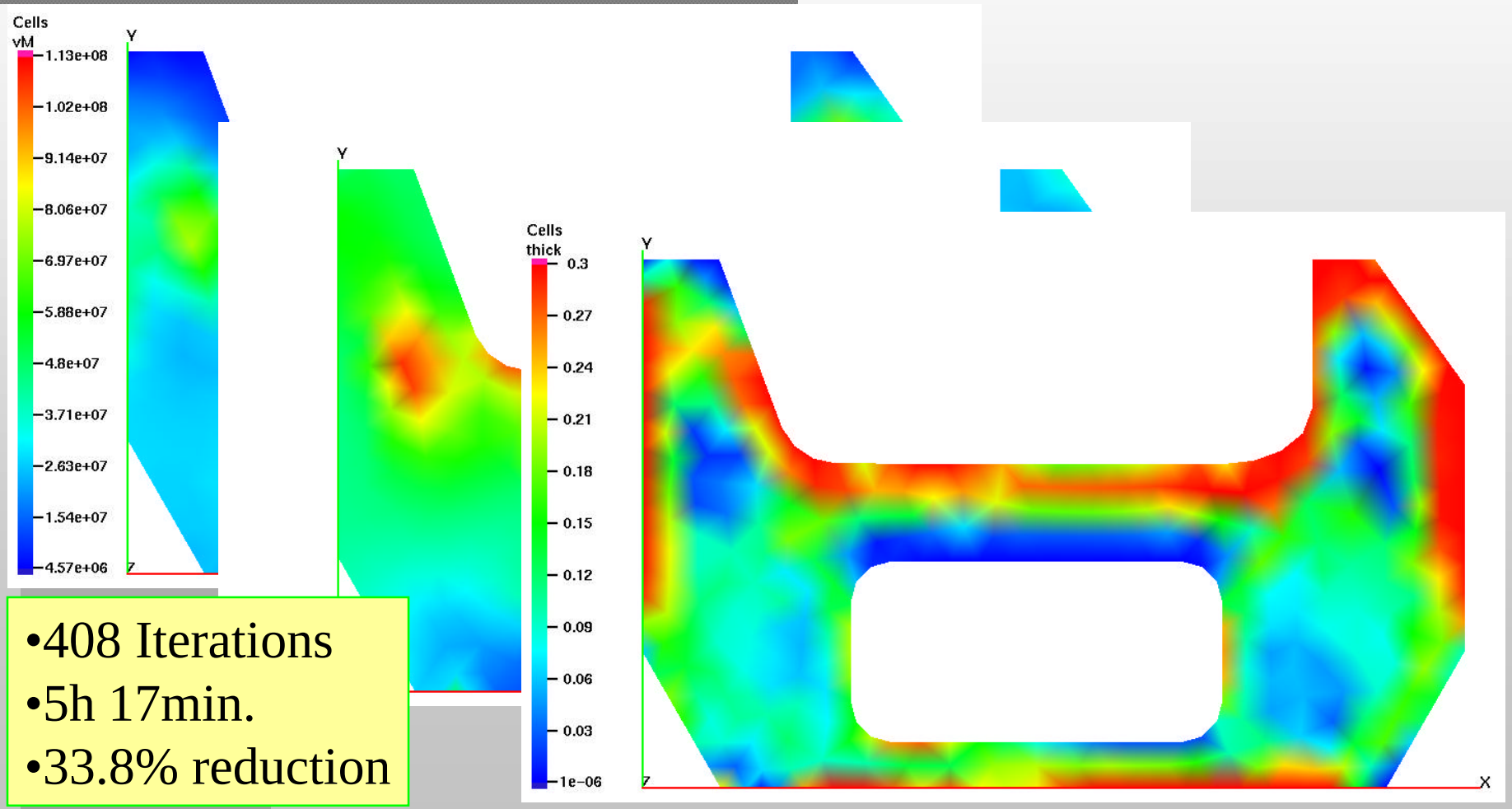


- 48 Iterations
- 11.5 min.
- 60.6% reduction

B-Spline C-frame (451 d.v.)

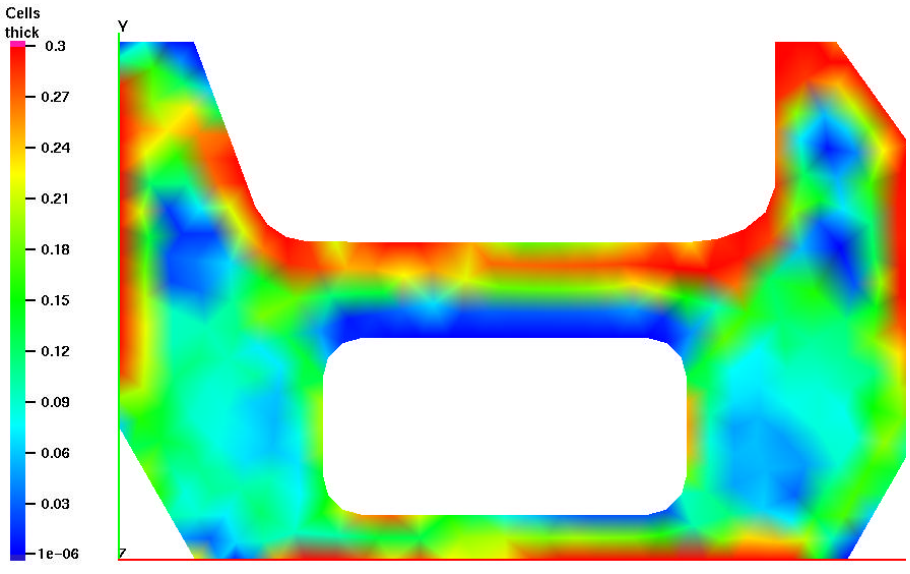


B-Spline C-frame (451 d.v.)

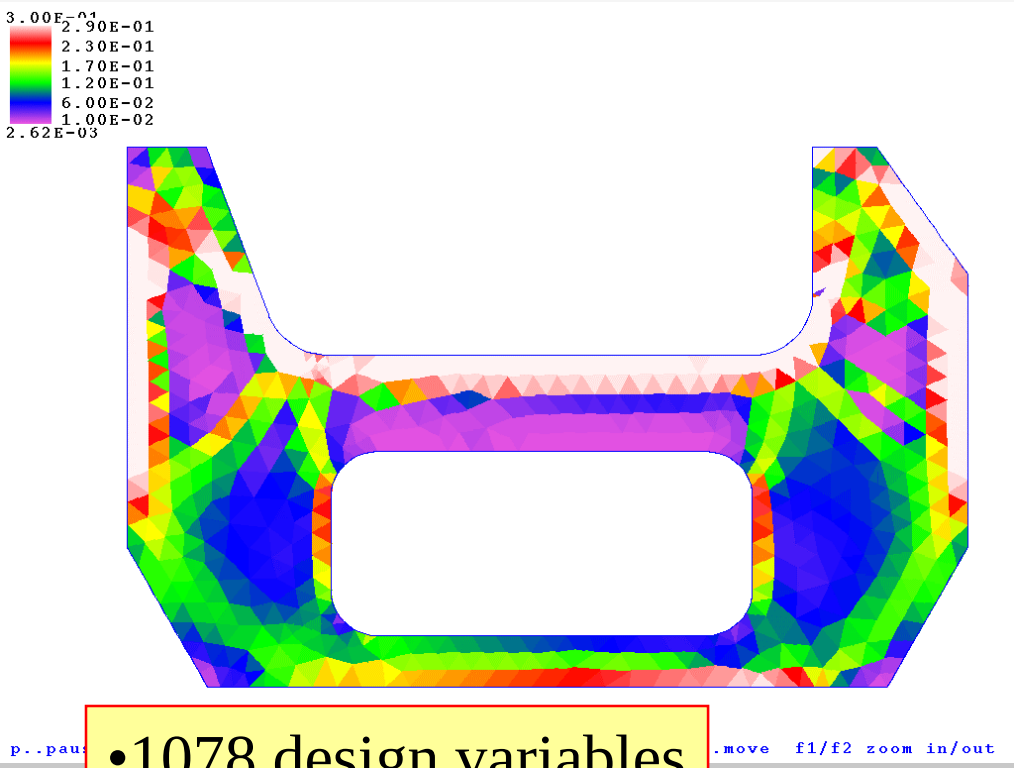


- 408 Iterations
- 5h 17min.
- 33.8% reduction

Cont. vs. discrete thickness



- 451 design variables
- 408 iterations
- 5h 17min.



- 1078 design variables
- 2200 iterations
- 105h

Optimization: Dreams?

- One **fast, adaptive** and **autom.** optimization code for
  - shape optimization
  - optimal sizing
  - (topology optimization)
- Requires competences in
  - optimization
  - computational geometry,mesh handling
  - finite elements
  - computer science